IMPLEMENTASI SWITCH OPENFLOW BERBASIS SOFTWARE DENGAN MEMANFAATKAN RASPBERRY PI UNTUK INFRASTRUKTUR SDN

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Imam Santoso
NIM: 135150307111049

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI SWITCH OPENFLOW BERBASIS SOFTWARE
DENGAN MEMANFAATKAN RASPBERRY PI
UNTUK INFRASTRUKTUR SDN

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Imam Santoso
NIM: 135150307111049

Skripsi ini telah diuji dan dinyatakan lulus pada
02 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Mochammad Hannats Hanafi Ichsan, S.ST., M.T.	Widhi Yahya, S.Kom., M.T.
NIK: 2014058812291001	NIK: 2016078911211001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 1 Juli 2018

Imam Santoso

NIM: 135150307111049

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah subhanahu wa ta'ala yang telah melimpahkan kasih dan sayang-Nya kepada kita, sehingga penulis bisa menyelesaikan skripsi dengan tepat waktu, yang kami beri Judul *"Implementasi Switch Openflow Berbasis Software Dengan Memanfaatkan Raspberry Pi Untuk Infrastruktur SDN".*

Tujuan dari penyusunan skripsi ini guna memenuhi salah satu syarat untuk bisa menempuh ujian sarjana pendidikan pada Fakultas Ilmu Komputer (FILKOM) Program Studi Teknik Informatika di Universitas Brawijaya Malang (UB).

Didalam pengerjaan skripsi ini telah melibatkan banyak pihak yang sangat membantu dalam banyak hal. Oleh sebab itu, disini penulis sampaikan rasa terima kasih sedalam-dalamnya kepada :

1. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang (FILKOM UB) yang telah memberikan ijin penelitian.
2. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya Malang (FILKOM UB) yang telah menyetujui permohonan penyusunan skripsi.
3. Bapak Agus Wahyu Widodo, S.T., M.Cs selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer.
4. Bapak Mochammad Hannats Hanafi Ichsan, S.ST, M.T. selaku Dosen Pembimbing I yang telah membimbing dalam penyusunan Skripsi ini hingga selesai.
5. Bapak Widhi Yahya, S.Kom., M.Sc. selaku Dosen Pembimbing II yang telah membimbing dalam penyusunan Skripsi ini hingga selesai.
6. Kedua orang tua serta keluarga penulis yang telah memberikan do'a serta dukungannya.
7. Sahabat dan rekan seperjuangan tercinta yang tiada henti memberi dukungan dan motivasi kepada penulis.
8. Semua pihak yang telah banyak membantu dalam penyusunan skripsi ini yang tidak bisa penulis sebutkan semuanya.

Akhir kata penulis mengucapkan mohon maaf atas kekurangan yang ada dalam laporan skripsi ini dan mengharapkan semoga laporan ini dapat bermanfaat bagi semua pihak.

Malang, 15 Juli 2018

Penulis
denmasmister@gmail.com

ABSTRAK

Software-Defined Networking (SDN) merupakan arsitektur yang dinamis, mudah dikelola, mudah beradaptasi dan *cost-efficient*. Switch openflow berbasis software pada Raspberry saat ini belum teruji performanya, sehingga masih belum dapat menjadi acuan yang tepat untuk dapat menggantikan perangkat *dedicated* switch openflow. Penelitian ini bertujuan untuk mengimplementasikan OpenvSwitch untuk diterapkan pada Raspberry sehingga menjadikannya sebagai switch openflow yang dapat digunakan didalam jaringan SDN dan kemudian menguji kemampuan dari Raspi switch, berupa *throughput, packet loss* dan *delay.* Hasil yang didapat dari penelitian kali ini bahwa Raspberry switch openflow dapat digunakan untuk menggantikan *dedicated* openflow switch pada skala kecil seperti lingkup perkantoran tetapi tidak untuk skala besar seperti perusahaan dikarenakan keterbatasan bandwidth pada USB to Ethernet Adapter, dan juga RAM dari Raspberry. Berdasarkan hasil pengujian *throughput, throughput* akan mengalami penurunan seiring dengan bertambahnya jumlah klien. Sedangkan berdasarkan hasil pengujian *packet loss, packet loss* akan naik secara signifikan ketika jumlah klien lebih dari 3. Kemudian berdasarkan pengujian *delay*, pengujian yang pertama memiliki *delay* yang lebih lama dibandingkan dengan pengujian selanjutnya, hal ini dikarenakan pada paket ICMP (ping) yang pertama, terjadi proses ARP sedangkan pada pengujian yang kedua dan seterusnya tidak melakukan proses ARP dikarenakan informasi mengenai ARP sudah ada pada ARP *cache*.

Kata kunci: sdn, openvswitch, software-based, raspberry, openflow

ABSTRACT

Software-Defined Networking (SDN) is a dynamic, manageable, adaptable and cost-efficient architecture. The Raspberry software-based openflow switch is currently untested, so it still can not be the right reference to replace dedicated openflow switch devices. This study aims to implement OpenvSwitch to be applied to Raspberries so as to make it an openflow switch that can be used within the SDN network and then test the performance of Raspi switches, throughput, packet loss and delay. The results obtained from this research that Raspberry openflow switches can be used to replace dedicated openflow switches on a small scale such as office scope but not for large scale companies like companies due to bandwidth limitation on USB to Ethernet adapters, as well as RAM from Raspberry. Based on the results of throughput testing, throughput will decrease as the number of clients increases. While based on packet loss test results, packet loss will increase significantly when the number of clients more than 3. Then based on delay testing, the first test has a longer delay compared to the next test, this is because in the first ICMP (ping) packet, the ARP process occurs while in the second and so on does not do the ARP process because the information about ARP already exists in the ARP cache.

Keywords: sdn, openvswitch, software-based, raspberry, openflow

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I PENDAHULUAN

1.1 Latar Belakang

Jaringan merupakan teknologi yang dewasa ini tidak dapat terlepas dari kehidupan masyarakat. Oleh sebab itu berbagai inovasi dalam rangka meningkatkan kualitas jaringan serta kepraktisan dalam hal installasi, *maintenance* dan *upgrade* perangkat kerap kali dilakukan. Pada infrastruktur jaringan tradisional (*traditional network*), setiap perangkat harus dikonfigurasi satu persatu ke masing-masing perangkat. Hal ini menambah kompleksitas dari jaringan saat topologi jaringan melibatkan perangkat yang lebih banyak, karena membutuhkan *re-configure* dan *re-policy* ke masing masing perangkat.

SDN adalah sebuah arsitektur jaringan dimana *network control* terpisah dari *forwading plane* dan SDN bersifat *programmable* (Sdxcentral, 2017). Bentuk *software defined network* (SDN) memungkinkan para peneliti, administrator dan operator untuk mengontrol jaringan mereka dengan perangkat lunak khusus dan menyediakan *Application Programming Interface* (API) terhadap tabel *forwarding* switch dari vendor yang berbeda.

Produk Switch Openflow buatan vendor sudah cukup banyak, namun masih memiliki kekurangan diantaranya membatasi inovasi dalam jaringan, menggunakan platform tetap dan sulit diujicobakan, serta arsitektur yang tertutup sehingga tidak bisa diprogram ulang. Selain itu Switch Openflow buatan vendor yang ada juga menawarkan harga yang mahal sehingga masih sulit untuk dapat dijangkau oleh kalangan instansi menengah kebawah (Rikie Kartadi, dkk. 2014).

Penelitian yang pernah dilakukan adalah penelitian untuk menguji performa implementasi *Software Based* OpenFlow berbasis Openwrt pada infrastruktur *Software Defined Network* dilakukan oleh Rikie Kartadie, dkk (2016). Rikie membahas bagaimana mengimplementasikan *Switch OpenFlow software based* dengan platform OpenWrt, kemudian dilakukan uji pembanding dengan menggunakan simulator Mininet. Hasil dari penelitiannya adalah performa *Switch OpenFlow software based* dapat dikatakan baik dengan hasil gap rata-rata pada setiap pengujian menunjukkan angka yang tidak tinggi, bahkan pada pengujian jitter dapat dikatakan sama (Rikie Kartadi, dkk. 2014).

Terdapat penelitian lain yang dilakukan oleh Rikie, yakni penggunaan Mikrotik RB750 untuk melakukan uji coba atas protokol OpenFlow, dimana MikroTik sudah support dengan *OpenFlow* semenjak versi 6.rc8 dan hasil pengujian menunjukan MikroTik dapat menjalankan protokol OpenFlow. Penggunaan arsitektur SDN dengan protokol OpenFlow tidak menghambat performa dari perangkat. Namun fungsi OpenFlow di MikroTik masih dibilang *experimental*, tidak untuk diproduksi masal (Rikie Kartadi, dkk. 2016).

Sementara itu, Raspberry Pi merupakan mini pc yang murah, bersifat *open source* dan bisa dimodifikasi sesuai kebutuhan penggunanya. Untuk penggunaan tingkat lanjut, Raspberry pi hampir tidak memiliki batasan. Banyak sekali kemungkinan pengembangan aplikasi yang dapat dilakukan dengan menggunakan Raspberry Pi (Raspberry Foundation, 2016).

Pada penelitian kali ini, berdasarkan latar belakang yang telah disampaikan, penulis tertarik untuk melakukan penelitian yang berjudul "IMPLEMENTASI SWITCH OPENFLOW BERBASIS SOFTWARE DENGAN MEMANFAATKAN RASPBERRY PI UNTUK INFRASTRUKTUR SDN" dengan perbedaan mendasar pada device yang digunakan, yaitu dengan mini komputer Raspberry pi dan menggunakan OpenVswitch sebagai *software* Openflow. Harapan dari penelitian ini adalah, pengujian dari Raspi switch akan didapatkan hasil *bandwidth, throughput, packet loss* dan *delay* yang cukup baik, dan bisa sebagai pengganti Switch Openflow *Hardware Based*.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah dipaparkan, rumusan masalah pada penilitian ini adalah sebagai berikut :

1. Bagaimana perancangan Switch Openflow *software-based* pada perangkat Raspberry pi?
2. Bagaimana implementasi Switch Openflow *software-based* pada perangkat Raspberry pi?
3. Bagaimana hasil yang didapatkan dari Switch Openflow *software-based* berbasis Raspberry pi?

1.3 Tujuan

Tujuan :

1. Merancang Switch Openflow *software-based* pada perangkat Raspberry pi.
2. Mengimplementasikan Switch Openflow *software-based* pada perangkat Raspberry pi.
3. Mengetahui hasil kinerja dari Switch Openflow *software-based* yang diterapkan pada Raspberry pi.

1.4 Manfaat

Manfaat penelitian ini adalah:

1. Mendalami pengetahuan tentang Switch OF *sofware based,* apa yang bisa dan tidak bisa dilakukan oleh Switch OF tersebut.
2. Dapat membuat sebuah *Switch OF* dengan biaya yang lebih hemat untuk menerapkan jaringan *Software Defined Network.*
3. Memberikan wawasan dan pengetahuan kepada pembaca terkait SDN, Switch OpenFlow*, Controller,* dan Raspberry pi.
4. Dapat dijadikan referensi untuk penelitian selanjutnya, seperti untuk analisis dan penggunaan USB to Ethernet card Gbit LAN sebagai switch OF.

1.5 Batasan masalah

Batasan masalah pada penelitian kali ini adalah:

1. Menggunakan maksimal dua buah Raspberry sebagai Switch OF dengan topologi linear.
2. Menggunakan Floodlight sebagai controller serta OpenFlow versi 1.3.0.

3. Maksimal (16 host secara paralel test) dengan 2 fisik host berupa virtual komputer untuk pengujian sistem yang dibangun.
4. USB to LAN Adapter yang digunakan memiliki Bandwidth *real* maksimal 5 Mbps walaupun didalam spesifikasinya tertulis *fast ethernet* 100 Mbps.

1.6 Sistematika pembahasan

Uraian mengenai pembahasan dari setiap bab adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini akan menguraikan tentang latar belakang yang menjadi alasan penelitian yang dilakukan. Rumusan masalah yang berisi pertanyaan penelitian, tujuan penelitian yakni tujuan yang akan dicapai pada penelitian, manfaat yang akan didapat dari dilakukannya penelitian ini, batasan masalah dari penelitian dan sistematika dari pembahasan dan penulisan penelitian "Implementasi Switch Openflow *Software Based* Dengan Memanfaatkan Raspbery-pi Untuk Infrastruktur SDN".

BAB II LANDASAN KEPUSTAKAAN

Dalam bab ini akan dibahas mengenai landasan teori dan referensi yang mendasari perancangan dan percobaan Swich OF *software-based* yang akan deterapkan pada Raspberry pi ini. Tinjauan pustaka dilakukan dengan merujuk pada penelitian terdahulu yang digunakan sebagai pembeda pada penelitian saat ini. Selain itu juga berisi mengenai definisi-definisi dan teori-teori yang menjadi dasar dalam penulisan penelitian serta penyusunan laporan yang diambil dari berbagai sumber.

BAB III METODOLOGI

Pada bab ini akan membahas metode yang digunakan dalam melakukan penelitian dan penyusunan skripsi ini. Metode pelaksanaan yang dilakukan selama pengerjaan skripsi ini meliputi studi dan pengkajian literatur, rekayasa kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian dan analisis hasil serta penarikan kesimpulan.

BAB IV REKAYASA KEBUTUHAN

Bab ini membahas tentang kebutuhan-kebutuhan pada sistem. Terdapat beberapa sub bab diantaranya, gambaran umum dari sistem yang akan dibuat, tujuan dari sistem, kegunaan, serta kebutuhan fungsionalitas yaitu kebutuhan yang mutlak harus terpenuhi agar sistem dapat berjalan sesuai dengan yang diharapkan, serta kebutuhan non fungsionalitas dari sistem.

BAB V PERANCANGAN DAN IMPLEMENTASI

Dalam bab ini dijelaskan tentang perancangan sistem yang terdiri dari perancangan perangkat keras, perancangan topologi dan perancangan perangkat lunak sebagai acuan dalam pembuatan sistem. Kemudian topologi menggunakan topologi linear yang akan ditambahkan penjelasan mengenai implementasi perangkat keras, dan selanjutnya akan dilakukan implementasi alat dari perancangan yang sudah dibuat.

BAB VI PENGUJIAN DAN ANALISIS

Pada bab ini memuat hasil pengujian dan analisis terhadap sistem yang telah dibangun dari switch openflow *software based* berbasis raspberry.

BAB VII PENUTUP

Bab ini membahas kesimpulan atas penelitian yang telah dikerjakan serta memberikan saran-saran untuk pengembangan lebih lanjut dari sistem yang telah selesai dibangun.

BAB II LANDASAN KEPUSTAKAAN

Bab ini berisi tinjauan pustaka yang meliputi kajian pustaka dan dasar teori yang diperlukan dalam melakukan penelitian. Kajian pustaka ini membahas penelitian yang sudah ada dan berkaitan dengan penelitian yang diusulkan, sedangkan dasar teori membahas berbagai teori yang diperlukan dalam menyusun penelitian yang diusulkan.

2.1 Tinjauan Pustaka

Pada penelitian ini, kajian pustaka diambil dari beberapa penelitian yang pernah dilakukan.

Penelitian pertama adalah penelitian untuk menguji performa implementasi *Software Based OpenFlow* berbasis Openwrt pada infrastruktur *Software Defined Network* dilakukan oleh Rikie Kartadie, dkk (2016). Rikie membahas bagaimana mengimplementasikan Switch OpenFlow *software based* dengan platform OpenWrt, kemudian dilakukan uji pembanding dengan menggunakan simulator Mininet. Hasil dari penelitiannya adalah performa Switch OpenFlow *software based* dapat dikatakan baik dengan hasil gap rata-rata pada setiap pengujian menunjukkan angka yang tidak tinggi, bahkan pada pengujian jitter dapat dikatakan sama.

Terdapat penelitian lain yakni penggunaan Mikrotik RB750 untuk melakukan uji coba atas protokol Openflow, dimana MikroTik sudah support dengan Openflow semenjak versi 6.rc8 dan hasil pengujian menunjukan Mikrotik dapat menjalankan protokol Openflow. Penggunaan arsitektur SDN dengan protokol Openflow tidak menghambat performa dari perangkat. Namun fungsi Openflow di Mikrotik masih dibilang *experimental*, tidak untuk diproduksi masal.

Dalam tesis Kartadie, R., melakukan penelitian dengan menggunakan dua buah switch OF *software-based* dan dibandingkan dengan *emulator* Mininet untuk melihat keberhasilan prototipe yang dibuat, namun belum diteliti tentang performa dari switch yang dikerjakan bila diimplementasikan ke infrastruktur berskala besar. Penelitian ini meneliti performa dari switch OF *software-based* dengan mengimplementasikannya pada topologi yang digunakan pada infrastruktur SDN.

Applement, M. dan De Boer, M. melakukan analisis performa terhadap *Hardware OpenFlow. Hardware* Openflow adalah seperti *NetFPGA card, Pica8* Openflow *on a Pronto switch* dan Open vSwitch. Pengujian dilakukan pada beberapa variabel diantaranya *QoS, Port Mirroring, fail over speed* dan *performance overhead*.

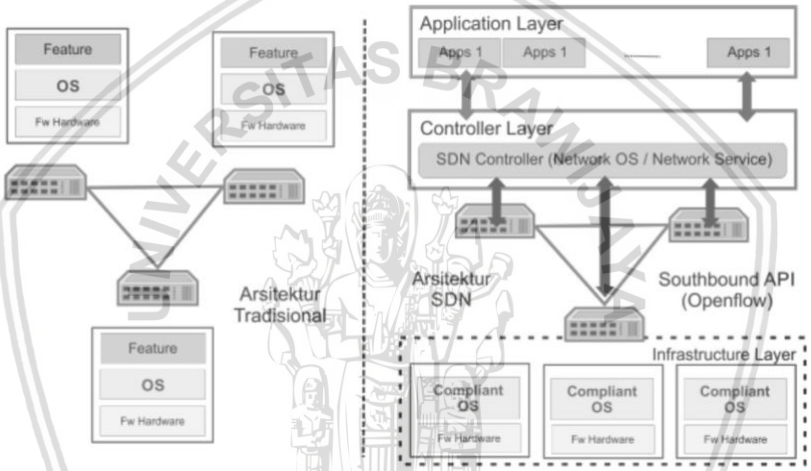
Perbedaan mendasar pada penelitian ini adalah dari jenis *hardware* yang digunakan, dalam penelitian kali jenis *hardware* yang digunakan adalah Raspberry pi. Kemudian hal mendasar lainnya adalah dari *software* Openflow yang digunakan, dimana dalam penelitian kali ini menggunakan *software* OpenVswitch.

2.2 Dasar Teori

Pada dasar teori, berisikan mengenai teori-teori dari komponen apa saja yang digunakan sebagai penunjang dalam penelitian. Dasar teori akan dibahas di subbab 2.2.1 sampai 2.2.7.

2.2.1 Software Defined Network

Konsep Sofware-Defined Network (SDN), pertama sekali diperkenalkan oleh Martin Casado di universitas stanford pada tahun 2007 dengan tulisan pada jurnalnya berjudul *"Ethane: Taking Control of the Enterprise"*. Pada jurnal tersebut dikatakan bahwa *ethane* adalah sebuah arsitektur baru untuk perusahaan, yang mengizinkan manajer mendefinisikan luasan dari sebuah jaringan, kebijakan jaringan dan kemudian menjalankannya secara langsung. *Ethane* adalah sebuah penyederhanaan yang ekstrim dari ethernet switch dengan sebuah kontroler terpusat yang mengatur hak masuk dan aliran *routing* (Khoerul Anam, dkk (2017).



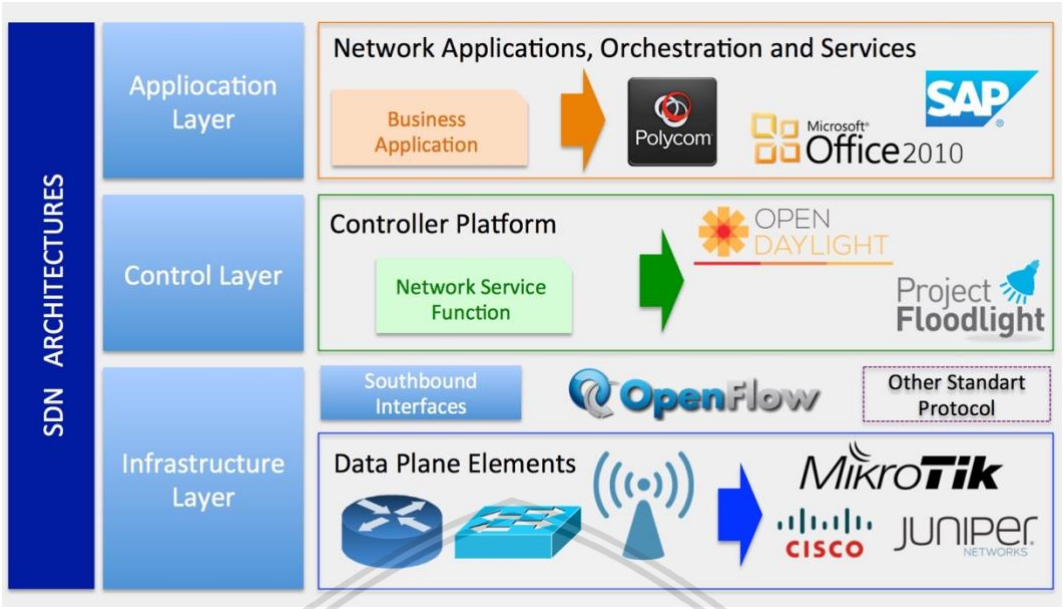
Gambar 0.1 Arsitektur SDN vs Arsitektur Tradisional

Sumber : Jurnal Khoerul Anam, dkk (2017)

Gambar 2.1 diatas menunjukan perbandingan arsitektur tradisional vs arsitektur SDN. *Software Defined Network* (SDN) adalah istilah yang merujuk pada konsep/paradigma baru dalam mendisain, mengelola dan mengimplementasikan jaringan, terutama untuk mendukung kebutuhan dan inovasi di bidang ini yg semakin lama semakin kompleks. Konsep dasar SDN adalah dengan melakukan pemisahan eksplisit antara *control* dan *forwarding plane*, serta kemudian melakukan abstraksi sistem dan meng-isolasi kompleksitas yg ada pada komponen atau sub-sistem dengan mendefinisikan antar-muka (*interface*) yg standard. "*Aris Cahyadi Risdianto, Muhammad Arif & Eueung Mulyana*".

2.2.2 Arsitektur SDN

Dalam konsep SDN, tersedia *open interface* yang memungkinkan sebuah entitas *software*/aplikasi untuk mengendalikan konektivitas yang disediakan oleh sejumlah sumber-daya jaringan, mengendalikan aliran trafik yg melewatinya serta melakukan inspeksi terhadap atau memodifikasi trafik tersebut. Gambar 2.2 berikut menunjukan arsitektur SDN beserta komponen dan interaksinya.



Gambar 0.2 Arsitektur SDN beserta komponen dan interaksinya
Sumber : Mikrotik ID – SDN implementation test on mikrotik (2016)

Arsitektur SDN dapat dilihat sebagai 3 lapis/bidang:

- **infrastruktur** (*data-plane* / *infrastructure layer*): terdiri dari elemen jaringan yg dapat mengatur SDN *Datapath* sesuai dengan instruksi yg diberikan melalui *Control-Data-Plane Interface* (CDPI)
- **kontrol** (*control plane* / *layer*): entitas kontrol (SDN *Controller*) mentranslasikan kebutuhan aplikasi dengan infrastruktur dengan memberikan instruksi yang sesuai untuk SDN *Datapath* serta memberikan informasi yang relevan dan dibutuhkan oleh SDN *Application*
- **aplikasi** (*application plane* / *layer*): berada pada lapis teratas, berkomunikasi dengan sistem via *NorthBound Interface* (NBI)

Bidang **Management & Admin** bertanggung-jawab dalam: inisiasi elemen jaringan, memasangkan SDN *Datapath* dengan SDN *Controller*, atau menkonfigurasi cakupan (*coverage*) dari SDN *Controller* dan SDN *App*.

Arsitektur SDN seperti dijelaskan di atas, dapat berjalan paralel dengan jaringan non-SDN, fitur yang sangat berguna untuk migrasi secara bertahap menuju jaringan SDN.

2.2.3 Switch Openflow

Switch OpenFlow tediri dari dua jenis, yang pertama adalah *hardware-based switch*, yang telah dijual secara komersial oleh beberapa vendor. Switch jenis ini telah memodifikasi *hardware*-nya, menggunakan TCAM (Ternary Content Addressable Memory) dan menggunakan OS khusus untuk mengimplementasikan *Flow-Table* dan protokol OpenFlow. Jenis yang kedua adalah *software-based switch* yang menggunakan sistem UNIX / Linux untuk mengimplementasikan seluruh fungsi Switch OpenFlow.

Dalam perangkat saklar tradisional (router switch), pengalihan paket dan perutean tingkat tinggi berada di perangkat yang sama. Sedangkan sebuah saklar OpenFlow memisahkan jalur data dari jalur kontrol. Pengontrol terpisah membuat

keputusan routing tingkat tinggi. Saklar dan pengontrol berkomunikasi dengan protokol OpenFlow.

Terdapat beberapa perbedaan mendasar dari switch komersial dengan Switch OpenFlow, dalam thesis Chung Yik,EE., perbedaan tersebut seperti terlihat pada tabel 2.1 dibawah ini.

Tabel 2.1 Tabel Perbandingan Switch menurut Chung Yik, EE

Switch Openflow	Switch komersial (Router switch)
Terpisahnya *control path* dan *data path*	*Control path* dan *data path* terletak pada perangkat yang sama (tidak terpisah)
Memungkinkan terjadi inovasi dalam jaringan	Membatasi inovasi dalam jaringan
Menyediakan *platform* yang dapat diteliti dan diujicobakan pada jaringan sesungguhnya.	Tetap dan sulit untuk diujicobakan (dibuat tetap oleh *vendor*)
Fungsi yang dapat didefiniskan oleh user (dapat diprogram)	Arsitektur tertutup sehingga tidak dapat diprogram ulang
Setiap keputusan untuk melakukan pengiriman dilakukan oleh kontroler	Mengirimkan semua paket yang diterima keluar dari switch

Sumber: Thesis Chung Yik, EE (2014)

2.2.4 Raspberry Pi

Raspberry Pi, sering disingkat dengan nama **Raspi**, adalah komputer papan tunggal (*single-board circuit*; SBC) yang seukuran dengan kartu kredit yang dapat digunakan untuk menjalankan program perkantoran, permainan komputer, dan sebagai pemutar media hingga video beresousi tinggi. Raspberry Pi dikembangkan oleh yayasan nirlaba, Rasberry Pi *Foundation*, yang digawangi sejumlah pengembang dan ahli komputer dari Universitas Cambridge, Inggris.

Ide dibalik Raspberry Pi diawali dari keinginan untuk mencetak pemrogram generasi baru. Seperti disebutkan dalam situs resmi Raspberry Pi *Foundation*, waktu itu Eben Upton, Rob Mullins, Jack Lang, dan Alan Mycroft, dari Laboratorium Komputer Universitas Cambridge memiliki kekhawatiran melihat kian turunnya keahlian dan jumlah siswa yang hendak belajar ilmu komputer. Mereka lantas mendirikan yayasan Raspberry Pi bersama dengan Pete Lomas dan David Braben pada 2009. Tiga tahun kemudian, Raspberry Pi Model B memasuki produksi massal. Dalam peluncuran pertamanya pada akhir Febuari 2012 dalam beberapa jam saja sudah terjual 100.000 unit. Pada bulan Februari 2016, Raspberry Pi Foundation mengumumkan bahwa mereka telah menjual 8.000.000.,- (juta) perangkat Raspi, sehingga menjadikannya sebagai perangkat paling laris di Inggris.

Raspberry Pi berjalan menggunakan sistem operasi *open source*, Linux. Raspberry Pi juga dapat dihubungkan dengan ke monitor komputer biasa, dan tambahan *port* untuk menghubungkannya dengan mouse dan keyboard. Sedangkan untuk penyimpanan data, Raspberry Pi tidak menggunakan Hardisk namun Raspberry Pi dapat menggunakan SD Card untuk menyimpan data, baik itu data *Operating System* ataupun untuk media penyimpanan data jangka panjang. Dengan memanfaatkan teknologi SoC (*system on chip*), Raspberry Pi berjalan di atas arsitektur ARM11 seperti yang dapat ditemui pada iPhone 3G maupun *smartphone* lain dan dilengkapi dengan videocore 4 GPU yang mampu memutar video dengan kualitas *BluRay*.

Kelebihan utama Raspberry Pi adalah ia dapat melakukan segala hal yang dapat dilakukan oleh komputer/laptop dengan sistem operasi Linux. Misalnya, membuat server, membuat program dengan berbagai macam bahasa, terutama bahasa tingkat tinggi seperti *Python*. Untuk fungsi sehari-hari, Raspberry dapat menjalankan sistem operasi berbasis GUI, sehingga dapat menggunakannya untuk melakukan pekerjaan standard seperti *browsing*, mendengarkan musik, nonton film, bermain game, mengetik dll. Untuk penggunaan tingkat lanjut, Raspberry pi hampir tidak memiliki batasan. Banyak sekali kemungkinan pengembangan aplikasi yang dapat dilakukan dengan menggunakan Raspberry pi.

2.2.4.1 Raspberry Pi 3 Model B

Raspberry pi 3 model B adalah generasi ketiga dari Raspberry pi dan menggantikan Raspberry pi 2 Model B pada bulan Februari 2016. Pada generasi ini telah dilakukan upgrade hardware, mulai dari jenis processor yang digunakan sampai dengan 1,2 GHz broadcom dengan lebar pita 64 bit, RAM yang ditanam hingga 1 GB, chipset, penambahan fitur bluetooth *low energy* (BLE *on board*), ethernet *port* hingga Gbits, USB *port* yang dimiliki juga bertambah hingga 4 *port*, penambahan *port* kamera CSP, port tampilan DSP, dan juga penambahan adapter *wireless* yang sudah ditanam pada *device*. Sumber daya USB Micronya pun naik sampai 2.5A. Berikut ini adalah gambar dari Raspberry 3 B (Raspberrypi.org (2017)).

Gambar 0.3 Raspberry Pi 3 Model B
Sumber : Raspberrypi.org (2017)

Spesifikasi dari Raspberry Pi 3 Model B adalah :

1. Quad Core 1.2GHz Broadcom BCM2837 CPU 64bit
2. RAM 1GB
3. BCM43438 LAN nirkabel dan *Bluetooth Low Energy* (BLE) *on board*
4. GPIO 40 pin
5. 4 port USB 2.0
6. 4 Pole stereo output dan port video komposit
7. HDMI *full size*
8. Port kamera CSI untuk menghubungkan kamera Raspberry Pi
9. Port tampilan DSI untuk menghubungkan display layar sentuh Raspberry Pi
10. Micro SD port untuk memuat sistem operasi dan menyimpan data
11. *Upgrade* sumber daya USB Micro naik sampai 2.5A

2.2.5 Open vSwitch

Open vSwitch adalah perangkat lunak multilayer yang berlisensi di bawah lisensi open source Apache 2. Selain itu, Open vSwitch merupakan proyek sumber terbuka yang memungkinkan *hypervisor* untuk virtualisasi lapisan jaringan. Open vSwitch ini melayani sejumlah besar mesin virtual yang berjalan pada satu atau lebih simpul fisik. Mesin virtual terhubung ke *port* virtual di jembatan virtual. Tujuannya adalah untuk mengimplementasikan platform switch kualitas produksi yang mendukung antarmuka manajemen standar dan membuka fungsi forwarding ke ekstensi dan kontrol terprogram (Openvswitch.org, 2018).

Open vSwitch sangat cocok untuk berfungsi sebagai saklar virtual di lingkungan VM. Selain mengekspos kontrol standar dan antarmuka visibilitas ke lapisan jaringan virtual, juga dirancang untuk mendukung distribusi di beberapa server fisik. Open vSwitch mendukung beberapa teknologi virtualisasi berbasis Linux termasuk Xen / XenServer, KVM, dan VirtualBox.

Sebagian besar kode ditulis dalam platform-independen C dan mudah dipindah ke lingkungan lain. Rilis Open vSwitch saat ini mendukung fitur-fitur berikut:

1. Model VLAN 802.1Q standar dengan port trunk dan akses
2. Ikatan NIC dengan atau tanpa LACP pada sakelar hulu
3. NetFlow, sFlow (R), dan mirroring untuk meningkatkan visibilitas
4. Konfigurasi QoS (Quality of Service), ditambah kepolisian
5. Geneve, GRE, VXLAN, STT, dan LISP tunneling
6. 802.1ag manajemen kesalahan konektivitas
7. OpenFlow 1.0 plus banyak ekstensi
8. Database konfigurasi transaksional dengan binding C dan Python
9. Penerusan kinerja tinggi menggunakan modul kernel Linux
10. Modul kernel Linux yang disertakan mendukung Linux 3.10 dan lebih tinggi.

Open vSwitch juga dapat beroperasi sepenuhnya di userspace tanpa bantuan dari modul kernel. Implementasi userspace ini harus lebih mudah ke port daripada switch berbasis kernel. OVS di userspace dapat mengakses perangkat Linux atau DPDK. Catatan Open vSwitch dengan datapath pengguna dan perangkat non DPDK dianggap eksperimental dan dilengkapi dengan biaya dalam kinerja.

Komponen utama dari distribusi ini adalah:

1. ovs-vswitchd, sebuah daemon yang mengimplementasikan switch, bersama dengan modul kernel Linux pendamping untuk switching berbasis aliran.
2. ovsdb-server, server basis data yang ringan yang memiliki pertanyaan-pertanyaan ovs-vswitchd untuk mendapatkan konfigurasinya.
3. ovs-dpctl, alat untuk mengkonfigurasi modul kernel switch.
4. Skrip dan spesifikasi untuk membangun RPM untuk Citrix XenServer dan Red Hat Enterprise Linux. The RPM XenServer memungkinkan Open vSwitch untuk diinstal pada host Citrix XenServer sebagai pengganti drop-in untuk switch-nya, dengan fungsi tambahan.
5. ovs-vsctl, utilitas untuk query dan memperbarui konfigurasi ovs-vswitchd.
6. ovs-appctl, utilitas yang mengirim perintah untuk menjalankan daemon Open vSwitch.

Open vSwitch juga menyediakan beberapa alat:

1. ovs-ofctl, utilitas untuk mempertanyakan dan mengendalikan switch dan pengontrol OpenFlow.
2. ovs-pki, utilitas untuk membuat dan mengelola infrastruktur kunci publik untuk switch OpenFlow.
3. ovs-testcontroller, pengendali OpenFlow sederhana yang mungkin berguna untuk pengujian (meskipun tidak untuk produksi).
4. Tambalan ke tcpdump yang memungkinkannya mem-parsing pesan OpenFlow.

2.2.6 Iperf

Iperf adalah alat baris perintah yang digunakan dalam mendiagnosis masalah kecepatan jaringan dengan mengukur *throughput* jaringan maksimum yang dapat ditangani oleh server. *Tools* ini sangat berguna ketika mengalami masalah kecepatan jaringan, karena dengan iperf kita dapat menentukan server mana yang tidak dapat mencapai *throughput* maksimum (Iperf.fr, 2018).

Iperf harus diinstal pada komputer di kedua ujung koneksi yang kita uji, karena satu sistem harus bertindak sebagai server, sementara yang lain bertindak sebagai klien. Klien akan terhubung ke server yang kita uji kecepatannya. Jika kita menggunakan sistem operasi berbasis Unix atau Linux pada komputer pribadi, kita dapat menginstal Iperf di komputer lokal.

2.2.7 Floodlight OpenFlow Controller

Floodlight *open* SDN *controller* adalah Openflow *controller* kelas enterprise, Apache-lisensi, berbasis Java (Projectfloodlight.org, 2018). Hal ini didukung oleh komunitas pengembang termasuk sejumlah insinyur dari Big Beralih Networks. Openflow adalah standar terbuka yang dikelola oleh *Open Networking Foundation*. Ini menentukan protokol melalui switch remote kontrol dapat memodifikasi perilaku perangkat jaringan melalui didefinisikan dengan baik "*set instruktion forwarding*". Floodlight dirancang untuk bekerja dengan meningkatnya jumlah switch, router, switch virtual, dan jalur akses yang mendukung standar OpenFlow. Beberapa fitur yang ditawarkan Floodlight:

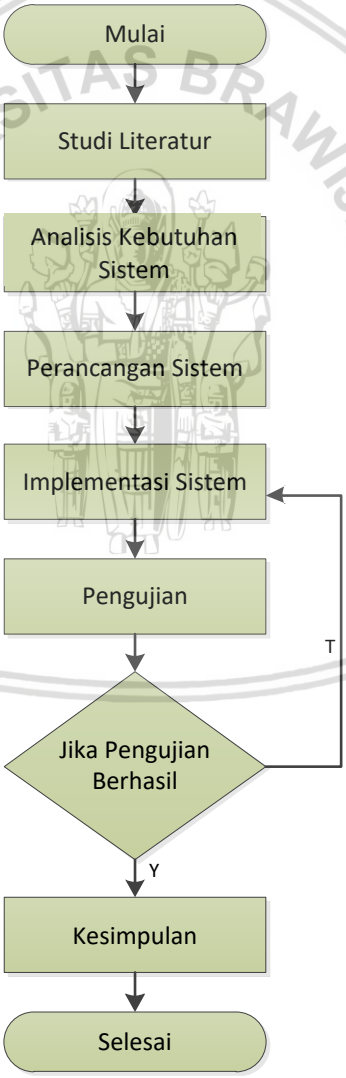
a. Menawarkan sistem modul beban yang membuatnya sederhana untuk memperluas dan meningkatkan.
b. Mudah untuk mengatur dengan dependensi minimal
c. Mendukung berbagai switch Openflow virtual dan fisik.
d. Dapat menangani OpenFlow dan non-OpenFlow jaringan campuran dapat mengelola beberapa " *islands* " dari Openflow switch *hardware* .
e. Dirancang untuk menjadi kinerja tinggi - adalah inti dari produk komersial dari Big Beralih Networks.
f. Dukungan untuk OpenStack (link) platform awan orkestrasi

BAB III METODOLOGI

Bab ini menjelaskan langkah-langkah yang akan ditempuh dalam penyusunan skripsi, meliputi studi dan pengkajian literatur, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian dan analisis serta penarikan kesimpulan.

3.1 Alur Metodologi Penelitian

Pada bagian ini dijelaskan tentang metode dan langkah-langkah yang dilakukan dalam pengerjaan penelitian ini. Penelitian ini bersifat implementatif karena berupa implementasi pembuatan sistem. Berikut merupakan tahapan-tahapan alur metodologi penelitian yang dilakukan secara umum dapat dilihat dari diagram alir pada Gambar 3.1.



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.2 Studi Literatur

Studi literatur merupakan tahapan untuk menyusun teori-teori dasar dan referensi yang mendukung dengan penelitian yang dilakukan. Ketika pelaksanaan penelitian ini dilakukan, banyak referensi yang mendukung dan dijadikan acuan baik dari jurnal, buku, maupun info yang didapatkan dari artikel yang ada di internet. Berikut ini merupakan beberapa dasar teori yang terkait dengan penelitian adalah sebagai berikut:

1. *Software Defined Network* (SDN)
2. Arsitektur SDN
3. Switch Openflow
4. Raspberry Pi 3
5. OpenvSwitch
6. *Floodlight controller*
7. *Iperf Network Testing Tools*

3.3 Analisis Kebutuhan Sistem

Analisis kebutuhan membahas mengenai kebutuhan apa saja yang dibutuhkan untuk melakukan analisis sistem pada penelitian yang dilakukan. Kebutuhan yang terkait dengan sistem ini akan dijelaskan sebagai berikut.

3.3.1 Kebutuhan Perangkat Keras (*Hardware*)

Analisa kebutuhan perangkat keras yaitu menganalisa kebutuha *hardware* apa saja yang digunakan untuk mewujudkan sistem pada penelitian ini. Adapun *hardware* yang digunakan, yaitu :

a. 1 buah Laptop spesifikasi tinggi
b. 2 buah Raspberry Pi 3 model B
c. 12 buah USB to LAN Adapter
d. 2 buah SD Card @16 GB Class 10
e. 10 buah *straight cable*
f. 2 unit Power Suply Raspberry Pi

3.3.2 Kebutuhan Perangkat Lunak (*Software*)

Analisis kebutuhan perangkat lunak (*software*) yaitu menganalisa kebutuhan perangkat lunak apa saja yang digunakan untuk mewujudkan sistem pada penelitian ini. Adapun *software* yang digunakan adalah sebagai berikut:

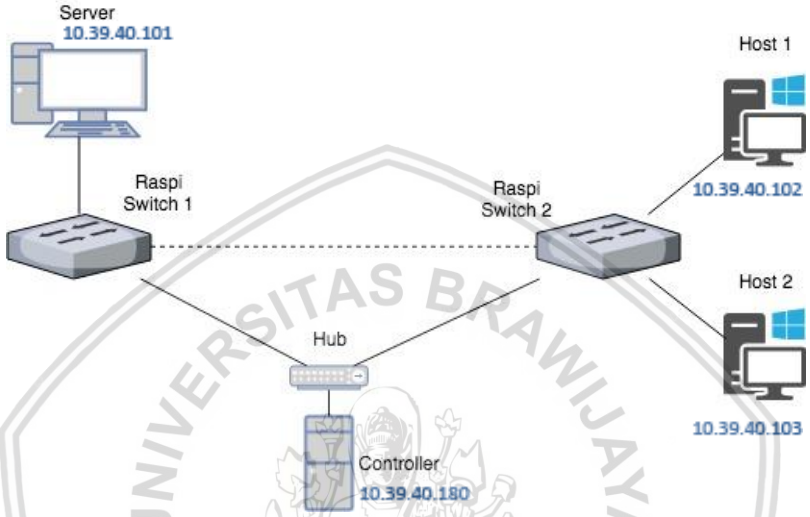
a. Raspbian 9 Streetch diinstall pada Raspi
b. Ubuntu 14.04 LTS diinstall pada sisi VM *controller*
c. Bunsenlabs yang diinstall pada VM server dan host
d. OpenvSwitch yang akan ditambahkan pada Raspi
e. Floodlight Controller yang akan diinstallkan pada sisi *controller*
f. Iperf yang diinstall pada VM server dan host

3.4 Perancangan Sistem

Perancangan sistem adalah tahap yang dilakukan setelah analisis agar pembuatan sistem dalam penelitian dapat dilakukan secarah terstruktur dan terarah. Perancangan sistem terdiri dari perancangan topologi dan perancangan hardware pada Raspi yang akan dijelaskan pada sub-bab dibawah ini.

3.4.1 Perancangan Topologi

Perancangan topologi sistem yang akan dibangun didalam skripsi ini adalah sebagai berikut.



Gambar 3.2 Perancangan Topologi Sistem

Pada gambar 3.2 diatas, garis lurus merupakan kabel/ line fisik sedangkan garis putus-putus merupakan logika karena setiap lan terhubung pada hub. Penggunaan hub diperlukan untuk topologi yang melibatkan dua buah switch openflow atau lebih dikarenakan pada sisi *controller* hanya memiliki sebuah port ethernet dan sebuah port untuk berkomunikasi, sedangkan pada topologi ini ada dua buah port ethernet dan dua buah port komunikasi yang mengharuskan untuk bisa berkomunikasi dengan *controller* pada port yang sama.

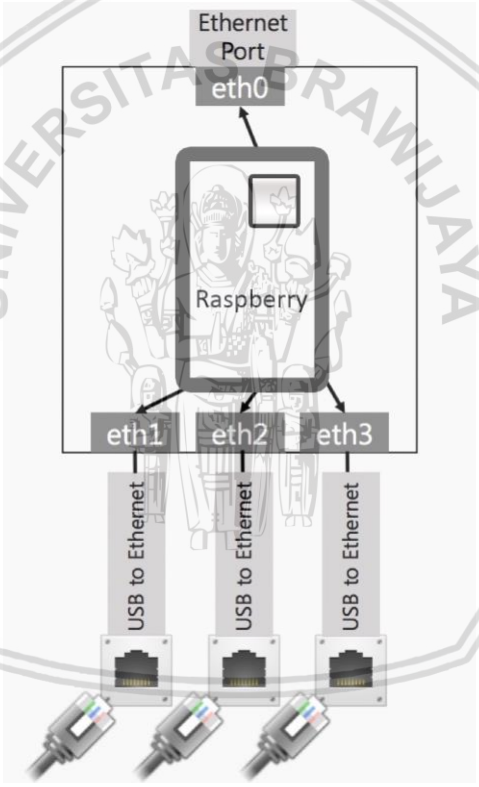
Berdasarkan pada gambar 3.2 diatas dapat dijabarkan proses kerjanya sebagai berikut:

1. Memastikan jaringan yang akan diimplementasikan SDN sudah berjalan dengan normal.
2. *Controller* ditambahkan dalam jaringan tersebut, dimana pada penelitian kali ini *controller* akan dijalankan pada sebuah *virtual computer*.
3. Setelah Raspberry Pi diaktifkan, dimana Raspberry Pi bertindak sebagai Switch Openflow, barulah *controller* dan perangkat bisa saling kenal dan *controller* ini akan secara otomatis berkenalan ke semua perangkat *(device provisioning system).*
4. Bila Openflow telah kita aktifkan maka *control plane* dan *data plane* di perangkat akan secara otomatis terpisah, dimana *control plane* berada pada *controller*nya sedangkan *data plane* berada diperangkat tersebut.

5. Selanjutnya, pada *controller* akan muncul yang namanya *virtual network*, yaitu akan muncul topologi yang menggambarkan setiap perangkat terhubung sekaligus dengan *controller*, dan *controller* akan mengenali interkoneksi antar switch.
6. Kemudian setelah itu ditambahkan *policy* dan *flow* pada *virtual* networknya. Maka pada *controller* akan melakukan *push configuration* ke setiap *device* yang ada dalam *network* tersebut. *Policy* dan *flow* inilah yang nantinya akan mengatur *traffic* pada jaringan.
7. *Controller* akan mengendalikan perangkat berdasarkan pada MAC Address, IP Address, atau TCP *Port* untuk melakukan aksi tertentu.

3.4.2 Perancangan Hardware Raspi

Perancangan hardware Raspi yang akan dibangun didalam skripsi ini adalah sebagai berikut.



Gambar 3.3 Perancangan Hardware Raspi

Pada gambar 3.3 diatas, dapat dijelaskan bahwa dikarenakan Raspi hanya memiliki 1 buah port ethernet, maka untuk dapat dijadikan sebagai switch openflow perlu ditambahkan beberapa modul USB to Ethernet. Dimana didalam penelitian ini masing masing Raspi akan ditambahkan 3 buah modul USB to Ethernet.

3.5 Implementasi

Tahapan implementasi dimulai dari installasi linux pada masing masing Raspberry Pi. Kemudian merakit seluruh hardware sesuai dengan perancangan

topologi dan perancangan hardware. Setelah itu, untuk dapat menjalankan Openflow maka perlu diinstall *software* bernama OpenvSwitch. Sebuah Openflow *Switch* tidak dapat bekerja tanpa adanya *controller,* maka ditambahkan pula *controller* didalam sebuah VM yang ditempatkan pada jaringan yang sama dengan menmanfaatkan mode *bridging network*.

3.6 Pengujian dan Analisis

Pengujian yang dilakukan pada penelitian ini terdiri dari pengujian terhadap sistem secara menyeluruh untuk dapat mengetahui kinerja sistem yang dibuat. Skenario pengujian yang dilakukan adalah dengan pengujian request dari klien yang berjumlah 3, 6, 12, 24, 48 sampai dengan 96 klien, sehingga akan didapatkan hasil untuk dilakukan analisis berupa *throughput, delay* dan *packet loss.* Pengujian *throughput* dilakukan melalui port TCP dengan menggunakan alat pengukur jaringan Iperf. Sedangkan *packet loss* dilakukan dengan menggunakan port UDP. Yang terahir adalah pengujian *delay* yang dilakukan dengan pengiriman paket ICMP (ping).

3.7 Penarikan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi, pengujian dan analisis sistem telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibuat. Isi kesimpulan diharapkan dapat menjadi acuan pada penelitian lain untuk pengembangan switch openflow *software-based*.

Tahap ini juga berisi saran untuk mengembangkan penelitian selanjutnya, diantaranya menambahkan perangkat keras, perangkat lunak ataupun mengganti metode yang digunakan agar sistem lebih sempurna.

BAB IV REKAYASA KEBUTUHAN

Bab ini menjelaskan tentang beberapa rekayasa kebutuhan daripada sistem yang harus dipenuhi ketika kita akan melakukan perancangan dan mengimplementasikan sistem, sehingga sistem yang dibangun dapat bekerja sesuai dengan hasil yang diharapkan.

4.1 Deskripsi Umum

Openflow merupakan protokol komunikasi *(communication protocol)* yang memberikan akses ke *forwarding* plane dari switch atau router melalui jaringan. Lingkungan kerja yang dibuat diharapkan agar Raspberry dapat menjadi switch openflow alternatif, dimana switch openflow ini bekerja dengan cara meneruskan paket yang dikontrol dari VM *controller*. Perangkat Raspberry hanya melakukan fungsi *forwarding* atau fungsi otot dan tidak melakukan fungsi *control plane* atau fungsi otak, karena untuk *control plane* itu hanya ada di *controller* sebagai fungsi otak tersebut.

Gambaran dari sistem adalah sebuah jaringan yang terdapat tiga buah Switch Openflow, satu terhubung ke Raspi Switch sebagai server dan memiliki dua buah host yang terhubung ke Raspi Switch yang lain. Switch Openflow disini adalah Switch Openflow *software-based* yang diterapkan pada mini komputer raspberry pi model 3 B. *Software* Openflow menggunakan OpenvSwitch, dan *cantroller* menggunakan floodlight. Jaringan dibentuk dengan menggunakan topologi linear. *Controller* dijankan melalui sebuah virtual machine linux ubuntu 14.04 LTS dengan fitur *bridge network*. *Host* juga dijalankan melalui virtual machine dengan type jaringan *bridge*, agar masing masing *host* seakan-akan berdiri sendiri dengan sebuah ethernet *port* yang terpisah.

4.2 Tujuan Sistem

Tujuan utama sistem dibangun adalah untuk mendapatkan hasil dari Raspberry yang berjalan sebagai swicth openflow sehingga dapat di implementasikan untuk jaringan SDN. Pada percobaan sistem ini diharapkan didapat hasil seperti tingkat keberhasilan sistem dan waktu yang dibutuhkan untuk melakukan transfer data, *bandwidth, throughput, packet loss* dan *delay*. Selain itu, dengan melakukan simulasi ini maka dapat diketahui kelebihan dan kekurangan untuk bahan pertimbangan penggunaan Raspberry switch OF.

4.3 Fungsionalitas Sistem

Sistem yang dibangun diharapkan dapat melakukan transfer data secara cepat sehingga dapat dijadikan alternatif switch *hardware* dan dapat menghemat energi serta biaya.

4.4 Rekayasa Kebutuhan Sistem

Pada bab rekayasa kebutuhan ini akan menguraikan semua kebutuhan yang memiliki tujuan agar switch OF *software-based* ini dapat bekerja sesuai

dengan tujuan awal yang terdiri dari kebutuhan fungsional dan non fungsional yang akan dibahas pada sub bab dibawah ini.

4.4.1 Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan yang harus dipenuhi agar suatu sistem dapat bekerja sesuai dengan tujuan. Apabila kebutuhan fungsional ini tidak terpenuhi maka sistem tidak berjalan dengan baik atau dapat dikatakan sistem dapat mengalami kegagalan.

1. Fungsi *Bridging Network* Pada MacBook

   Fungsi ini mengharuskan modul USB to Ethernet adapter dapat terbridge, dan dibaca oleh macbook yang kemudian akan direpresentasikan sebagai fisik ethernet port dari masing-masing virtual *host* dan *controller*, yang akan menjadi *port* komunikasi antar switch, antar *host*, dan antar switch dengan *host*.

2. *Controller* Dapat Menerima Data dan Menampilkan Kondisi Realtime Melalui Web Ui

   Antarmuka pada sistem akan ditampilkan pada sisi web ui *controller* floodlight. Antarmuka ini digunakan untuk melihat topologi yang sedang berjalan, switch OF, dan host yang sedang tersambung didalam jaringan.

3. Fungsi *Policy* dan *Flow*

   Fungsi ini digunakan agar *controller* supaya dapat melakukan *push configuration* ke setiap device yang ada dalam network tersebut. *Policy* dan *flow* inilah yang nantinya akan mengatur *traffic* pada jaringan.

4.4.2 Kebutuhan Non Fungsional

Pada tahap ini dilakukan analisis kebutuhan non fungsional dari sistem, sehingga dapat diuraikan sebagai berikut.

1. Menggunakan floodlight sebagai *software controller* yang digunakan sebagai otak dari switch OF.
2. Prototipe alat pada sistem switch OF didesain se-familiar mungkin untuk memudahkan pengguna.
3. Prototipe alat pada sistem didesain secara portable yang memudahkan untuk mobilitas.
4. Memperlihatkan kemungkinan kesulitan yang dihadapi oleh user. Salah satunya dengan pertimbangan pembuatan topologi yang tidak terlalu rumit.

4.5 Batasan Desain Sistem

Dalam pembuatan Switch Openflow dengan menggunakan Raspberry ini terdapat beberapa batasan yang diterapkan sehingga dalam proses pembahasan, perancangan maupun implementasi dari sistem ini tidak mencakup jangkauan yang luas. Batasan-batasan dari desain sistem tersebut antara lain :

1) Menggunakan maksimal 2 buah Raspberry sebagai Switch OF dengan topologi linear.

2) Menggunakan Floodlight sebagai controller serta OpenFlow versi 1.3.0.
3) Maksimal (16 host secara paralel test) dengan 2 fisik host berupa virtual komputer untuk pengujian sistem yang dibangun.
4) USB to LAN Adapter yang digunakan memiliki Bandwidth maksimal 5 Mbps

BAB V PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai perancangan dan implementasi sistem sesuai hasil dari analisis kebutuhuhan dimulai dari perancangan topologi percobaan, perancangan switch openflow yang akan digunakan dan implementasi sistem, instalasi serta konfigurasi sistem.

5.1 Perancangan Sistem

Pada bagian ini akan dijelaskan secara bertahap dimulai dengan spesifikasi alat yang digunakan, gambaran garis besar mengenai sistem yang akan dirancang dan ringkasan rancangan sistem secara keseluruhan.

5.1.1 Spesifikasi Alat

Dalam pelaksanaan tugas akhir ini, menggunakan beberapa alat bantu komputasi akan digunakan, baik berupa perangkat keras (*hardware*) maupun perangkat lunak (*software*), sebagai berikut:

a. Perangkat keras (*hardware*)

1. MacBook Pro (Retina, 15-inch, Mid 2014), dengan spesifikasi processor 2,5 GHz Intel Core i7, RAM 16 GB 1600 MHz DDR3, Kartu grafis Intel Iris Pro 1536 MB, SSD 500 GB.
2. 2 Paket Raspberry Pi 3B, dengan spesifikasi CPU: 4x ARM Cortex-A53, 1.2GHz, GPU: Broadcom VideoCore IV, RAM: 1GB LPDDR2 (900 MHz), Networking: 10/100 Ethernet, 2.4GHz 802.11n *wireless*, MicroSD Sandisk 16GB Class 10.
3. 10 buah USB to Ethernet *Adapter*.
4. Switch Hub 8 port *Fast Ethernet* TP-Link
5. 10 buah kabel UTP Cat 6 (*straight*)

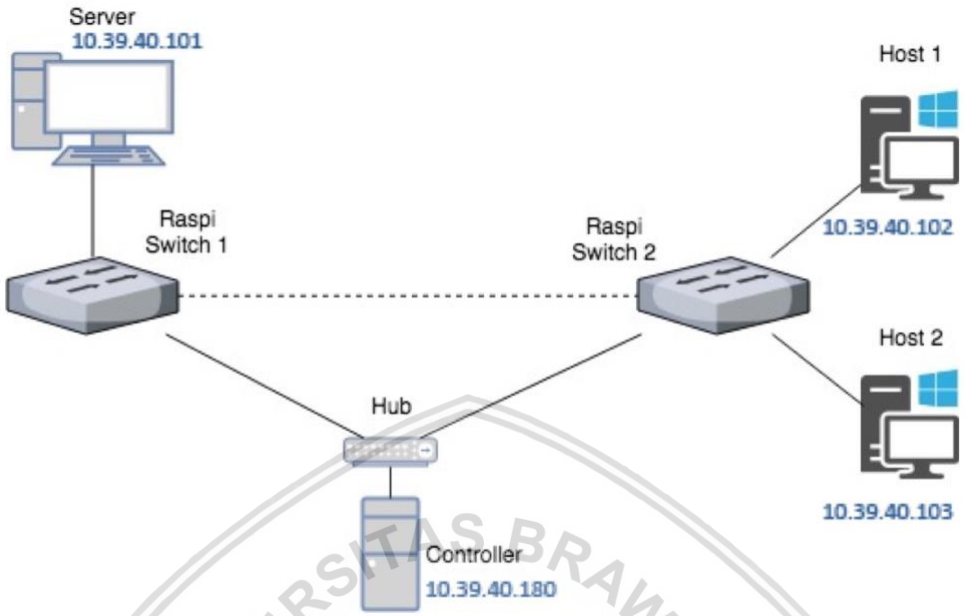
b. Perangkat lunak (*software*)

1. Raspbian OS
2. OpenVswitch v.2.3x.
3. Ubuntu 14.04 LTS
4. Floodlight controller
5. Mininet
6. Virtual Box
7. *Bunsenlabs linux*
8. *Iperf Network Tools*
9. *Bandwidth Monitoring Tools*

Didalam macbook tersebut diatas saya menjalankan 3 buah virtual machine, yakni sebagai *controller*, *server*, *host* 1 dan *host* 2. *Controller* memiliki spesifikasi CPU 1 Core 2,54 GHz Core i7, RAM 2 GB dan HDD 8 GB, sedangkan *host* memiliki spesifikasi CPU 1 Core 2,54 GHz Core i7, RAM 1 GB, memori *swap* 2 GB dan HDD 5 GB. Spesifikasi tersebut sangat cukup dan ringan untuk menjalankan OS didalamnya.

5.1.2 Topologi Percobaan

Topologi percobaan pada penelitian ini digambarkan sebagai berikut.



Gambar 5.1 Topologi Percobaan

Pada gambar 5.1 diatas terdiri dari 1 buah *virtual computer* sebagai *controller*, 2 buah raspi sebagai switch OF *software-based* dan 2 buah *virtual desktop* sebagai *host*.

Penelitian ini secara umum terdiri dari beberapa langkah, yaitu pengujian performa switch OF *software-based* dengan 6 skenario, dimulai dari pengujian dengan 3 buah klien, 6, 12, 24, 48 hingga yang terakhir adalah 96 klien.

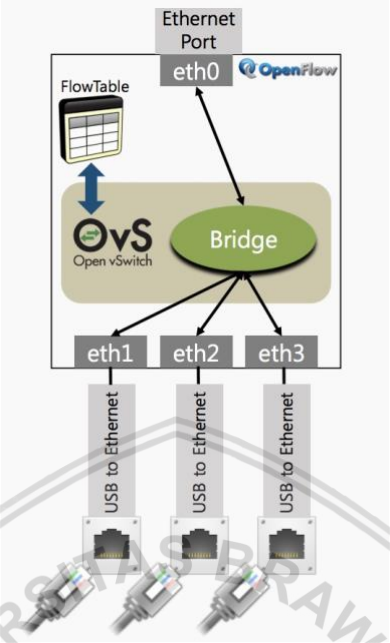
5.1.3 Perancangan Switch Openflow Pada Raspi

Untuk dapat menjadikan Raspberry sebagai switch openflow maka harus ditambahkan sebuah *software* openflow, dan didalam penelitian kali ini penulis menggunakan OpenvSwitch sebagai *software* Openflow. Skema dari openflow yang akan dibangun dengan raspberry akan digambarkan seperti pada gambar 5.2 dan 5.3 dibawah ini.

Gambar 5.2 Skema Perancangan Switch Openflow

Pada gambar 5.2 diatas menunjukkan alur/proses untuk menjadikan Raspi sebagai switch OF *software-based* yakni dengan menambahkan OpenvSwitch

didalamnya. Sedangkan gambar 5.3 dibawah ini akan menjelaskan desain diagram sistem dari raspberry switch OF.



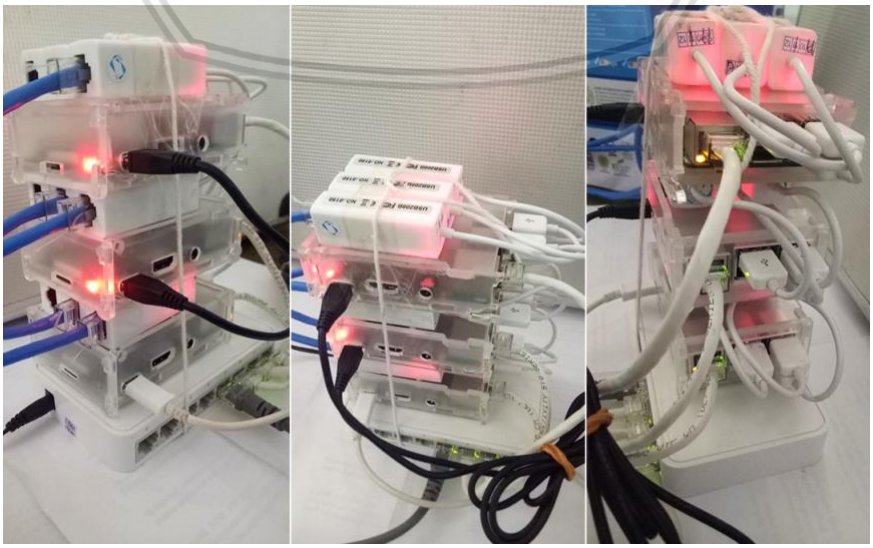
Gambar 5.3 Diagram Sistem Raspberry

5.2 Implementasi Sistem

Implementasi sistem switch OF *software-based* ini dilakukan berdasarkan subbab 3.4 setelah tahab analisis kebutuhan perangkat keras dan perangkat lunak. Selanjutnya adalah perancangan perangkat keras, installasi perangkat lunak dan konfigurasi serta implemantasi sistem.

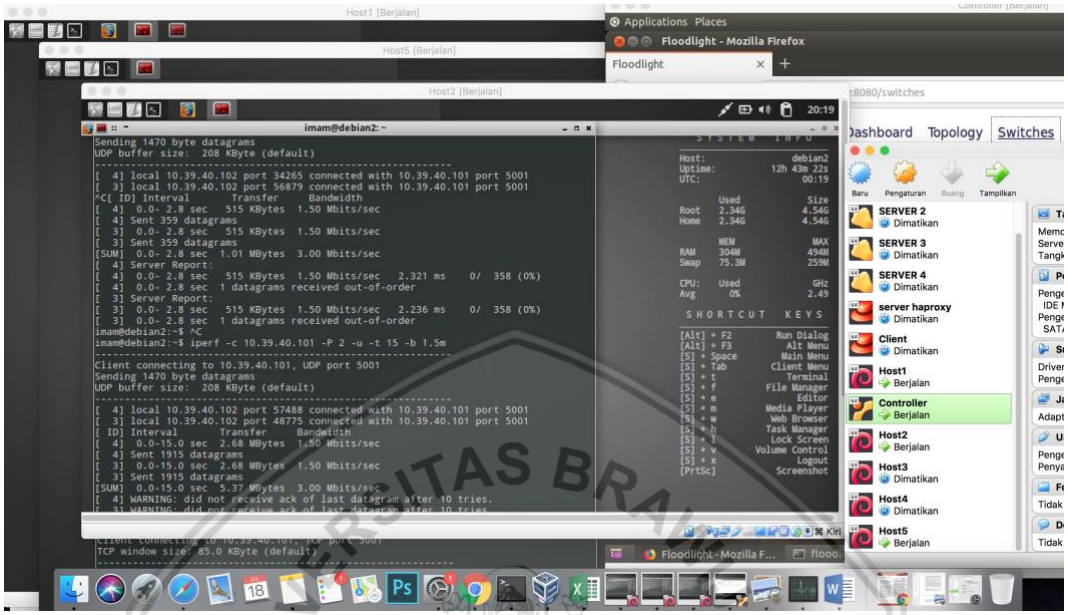
5.2.1 Implementasi Perangkat Keras

Perangkat keras yang digunakan untuk membangun sistem terdiri dari dua set Raspberry pi 3B, MacBook, USB to Ethernet adapter, kabel jaringan dan hub. Alat tersebut dirakit seperti gambar 5.4 dibawah ini.



Gambar 5.4 Tampilan Prototype Sistem

Gambar 5.4 diatas merupakan gambar perangkat keras prototipe dari sistem yang telah dibuat, dimana semua komponen sudah terhubung menjadi satu kesatuan. Selanjutnya adalah *virtual host* yang ditempatkan dalam sebuah macbook seperti gambar 5.5 dibawah ini.



Gambar 5.5 Tampilan Virtual Machine

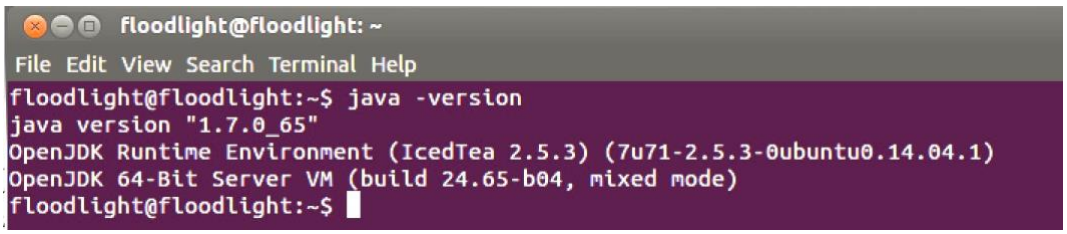
5.2.2 Implementasi Perangkat Lunak

Lingkungan perangkat lunak yang digunakan untuk mengimplementasikan sistem ini adalah sebagai berikut:

1. Sistem operasi Ubuntu 14.04 sebagai OS *controller*.
2. OpenvSwitch sebagai software openflow
3. Sistem operasi Bunsenlabs sebagai *host*.
4. Floodlight sebagai *controller*
5. Iperf dan Bandwidth monitor sebagai alat uji jaringan.

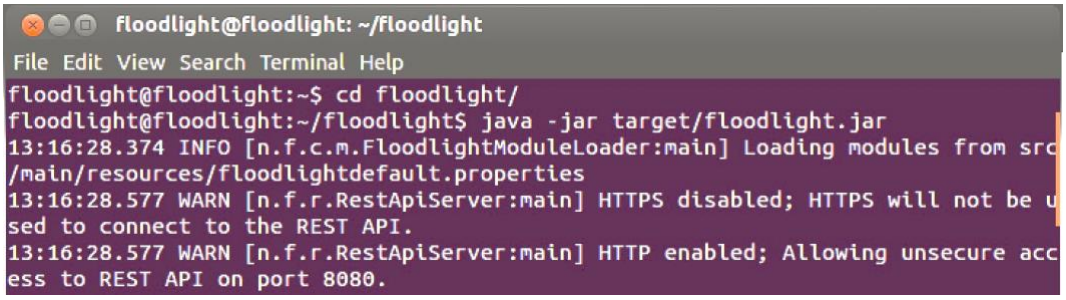
5.2.3 Installasi Floodlight

Sebelum melakukan installasi floodlight, dibutuhkan beberapa persyaratan *module* yang harus terinstal, diantaranya adalah *java development kit* dan *ant* atau *maven* untuk mengompile. Sebelumnya pastikan bahwa VM telah terhubung ke jaringan internet. Untuk melihat apakah sudah berhasil bisa melakukan dengan perintah sebagai berikut yang akan ditampilkan pada gambar 5.6 berikut ini.

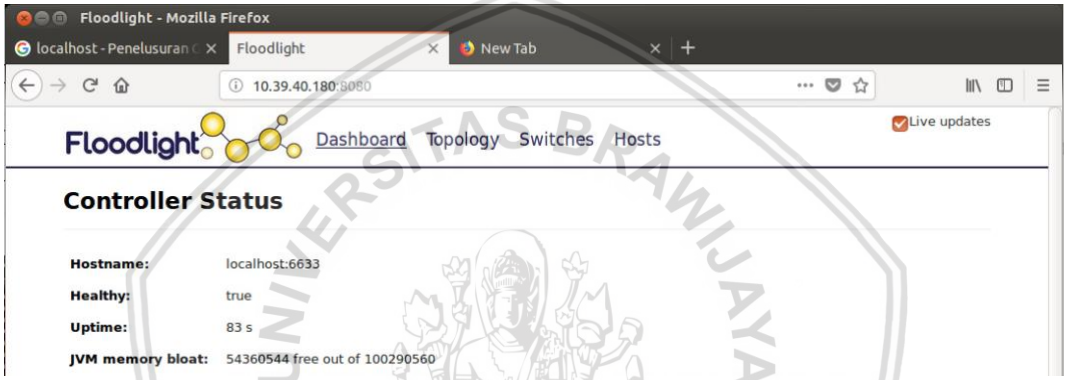


Gambar 5.6 Java Version

Selanjutnya melakukan installasi floodlight melalui Scratch. Kemudian bila sudah selesai kita bisa menjalankan floodlight dengan cara seperti pada gambar 5.7 dan 5.8.



Gambar 5.7 Gambar proses ketika floodlight dijalankan

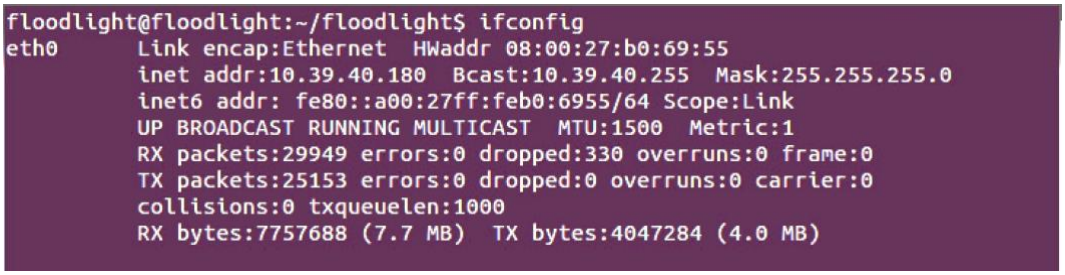


Gambar 5.8 Gambar web ui floodlight

Web UI Floodlight dapat diakses dengan mengetikkan pada *url browser <ip address controller:8080/ui/index.html>*. Ditampilan web ui tersebut terlihat kosong, karena belom ada ovs yang terhubung ke *controller*.

5.2.4 Konfigurasi Ip Address

Setelah melakukan installasi sistem operasi ubuntu 14.04 LTS yang berdiri di dalam sebuah *virtual machine*, kemudian juga telah sukses terinstall floodlight *controller* terlebih dahulu dilakukan konfigurasi pengelamatan *ip static* agar nantinya mudah untuk *set-controller* pada ovs *system*. *Ip address* yang diset hanya untuk eth0 saja, dimana eth0 ini nantinya akan dihubungkan pada sebuah switch *hardware* agar bisa berjalan *multiple ovs* dengan sebuah *controller*. Ip 10.39.40.180 /24. Kemudian *restart* jaringan dan cek konfigurasi dengan *ifconfig* pada command line seperti yang ditunjukkan pada gambar 5.9 dibawah ini.



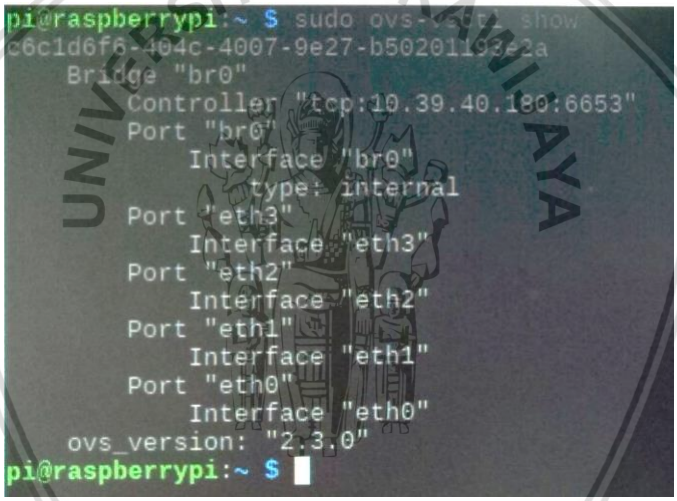
Gambar 5.9 Gambar ip interfaces *controller*

5.2.5 Konfigurasi OVS Pada Pi

Untuk dapat menjalankan Raspberry Pi sebagai switch openflow maka dibutuhkan sebuah *software openflow*. Salah satu openflow yang terbaik adalah OpenVswitch. Pertama pastikan raspberry mendapatkan koneksi internet, kemudian lakukan konfigurasi senyaman mungkin pada tampilan desktop, resolusi layar, keyboard layout, fitur fitur yang ingin diaktifkan atau dimatikan pada raspberry. Hal tersebut diatas akan meminimalisir error dan hal hal yang tidak diinginkan pada proses selanjutnya.

Setelah Open vSwitch terinstall dengan benar maka *bridge* br0 dapat ditambahkan. *Bridge* tersebut akan bekerja sebagai *virtual switch* yang akan meneruskan paket berdasarkan *flow-table* yang diberikan.

Setelah *bridge* selesai ditambahkan, dapat dilakukan konfigurasi versi OpenFlow yang digunakan serta menambahkan *controller* pada *switch*. *Controller* ditambahkan dengan cara memberikan alamat ip dari *controller* serta protokol yang digunakan untuk komunikasi. Secara *default port* yang digunakan oleh *controller floodlight* adalah tcp:6653. Gambar 5.10 dibawah ini menunjukkan kondisi ketika ovs belum terhubung ke *controller*.

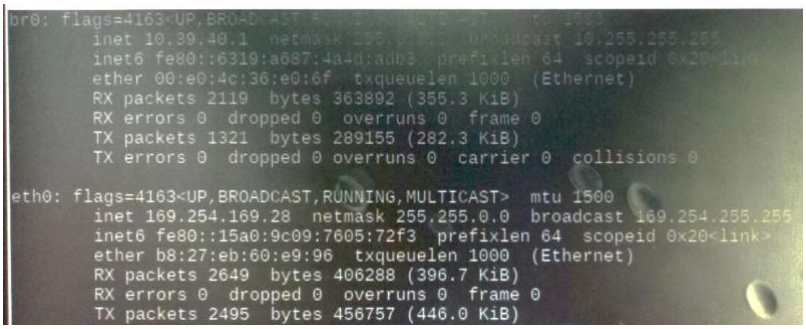


Gambar 5.10 Gambar info port pada ovs sebelum terkoneksi

Setelah itu untuk bisa terhubung ke *controller* kita harus menyeting konfigurasi *ip static* pada <BridgeName>. Perlu diingat bahwa sebagaimana fungsi switch hanya untuk *forwarding*, maka biarkan kosong *ip address* pada seluruh eth *port* karena switch tidak mungkin memiliki ip address. Kemudian lakukan setting *ip address* melalui *dhcpcd.conf* bukan */interfaces* dengan IP 10.39.40.1/24.

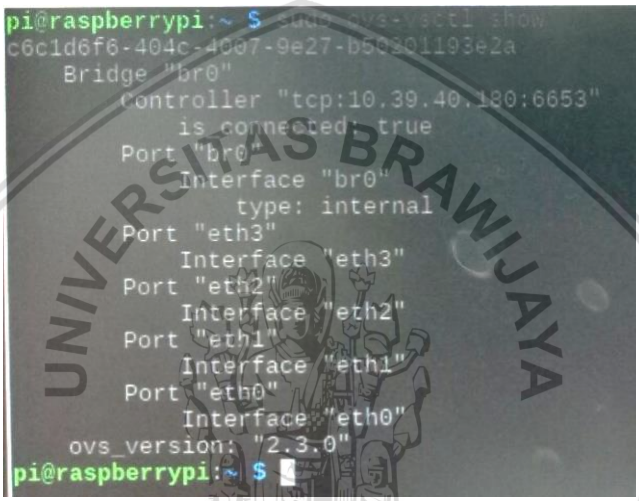
Kemudian lakukan hal yang sama pada switch ovs yang lain dengan *host* address selain .1 dan .180 (host tersebut sudah digunakan oleh ip ovs1 dan ip controller). Bila sudah, lakukan restart jaringan.

Selanjutnya, kita bisa melihat kembali ip dari konfigurasi yang telah kita buat, maka hasilnya akan tampak seperti pada gambar 5.11 berikut. Dari gambar tersebut menunjukkan bahwa ip dari br0 atau ip dari ovs1 sudah masuk yakni 10.39.40.1/24. Adapun eth0, eth1 dan seterusnya muncul ip 169.254.x.x biarkan saja. Hal tersebut tidak berpengaruh sama sekali.



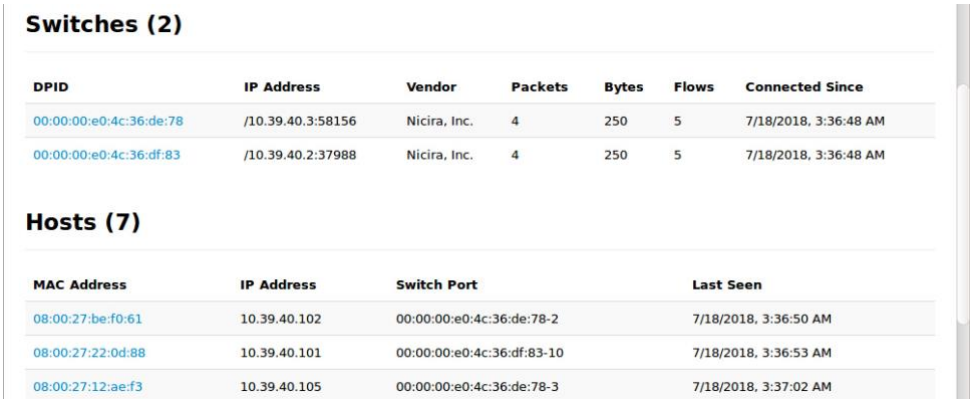
Gambar 5.11 Gambar Konfigurasi ovs1

Selanjutnya apabila kita melihat pada port ovs, maka status *controller* akan terhubung <is connected: true> sebagaimana pada gambar 5.12 berikut.



Gambar 5.12 Gambar status ovs yang terkoneksi

Untuk menambahkan port disesuaikan dengan jumlah modul USB to Ethernet Adapter yang dipasang pada Raspi. Perintah yang dilakukan adalah ovs-vsctl add-port <nama bridge> eth<n> — add-port br0 eth <n>. List device yang terhubung didalam jaringan SDN bisa dilihat melalui *controller* web ui floodlight sebagai berikut.



Switches (2)

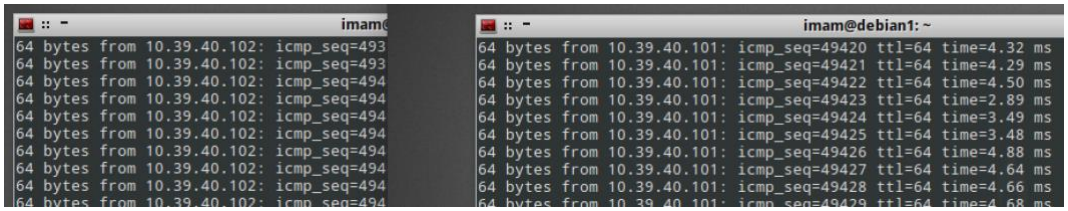
DPID	IP Address	Vendor	Packets	Bytes	Flows	Connected Since
00:00:00:e0:4c:36:de:78	/10.39.40.3:58156	Nicira, Inc.	4	250	5	7/18/2018, 3:36:48 AM
00:00:00:e0:4c:36:df:83	/10.39.40.2:37988	Nicira, Inc.	4	250	5	7/18/2018, 3:36:48 AM

Hosts (7)

MAC Address	IP Address	Switch Port	Last Seen
08:00:27:be:f0:61	10.39.40.102	00:00:00:e0:4c:36:de:78-2	7/18/2018, 3:36:50 AM
08:00:27:22:0d:88	10.39.40.101	00:00:00:e0:4c:36:df:83-10	7/18/2018, 3:36:53 AM
08:00:27:12:ae:f3	10.39.40.105	00:00:00:e0:4c:36:de:78-3	7/18/2018, 3:37:02 AM

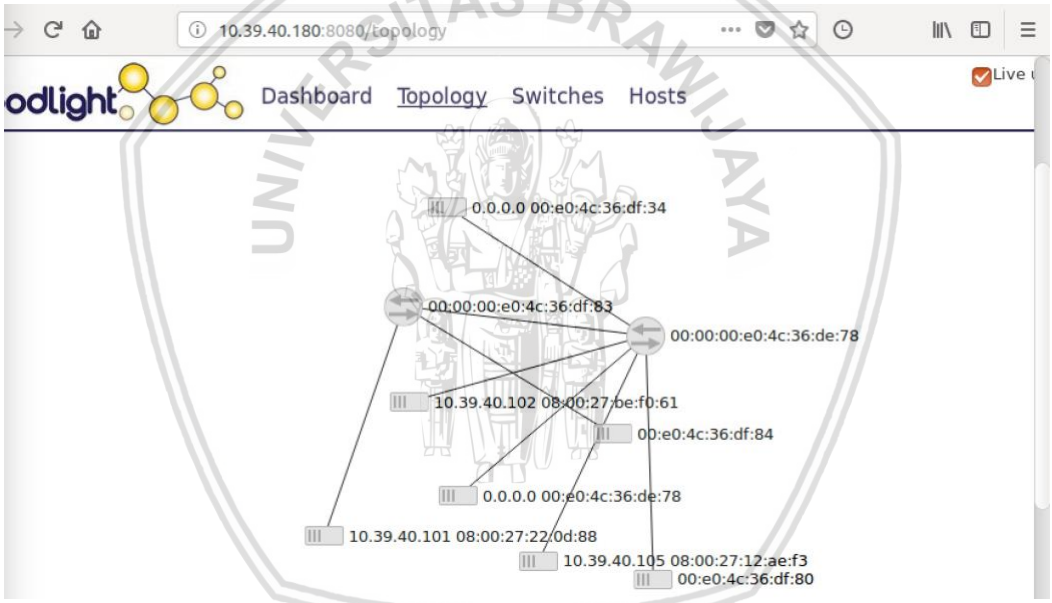
Gambar 5.13 Gambar list device yang berada didalam jaringan SDN

Dari gambar 5.13 diatas dapat kita lihat bahwasanya *device* yang sedang terhubung pada jaringan SDN terdapat dua buah switch ovs, pada ip 10.39.40.1 dan 10.39.40.2 kemudian terdapat 2 buah host fisik dengan ip address 10.39.40.101 dan 10.39.40.102. Status kedua host sudah bisa terkoneksi dengan baik yang ditunjukkan pada gambar 5.14 dibawah ini.



Gambar 5.14 Gambar koneksi antar dua buah host

Kita juga bisa melihat visual topologinya seperti apa pada tab topology, sebagaimana pada gambar 5.15 dibawah ini.



Gambar 5.15 Gambar Topologi Jaringan

5.2.6 Installasi Iperf

Untuk dapat mengukur *throughput* dan *packet loss*, penulis memilih *Iperf* sebagai *tools* untuk melakukan pengukuran. Penginstallan dapat dilakukan dengan <*sudo apt-get install iperf*>.

BAB VI PENGUJIAN & ANALISIS HASIL

Pada bab ini memuat hasil pengujian dan analisis terhadap sistem yang telah dibangun dari Raspi switch OF. Pengujian akan dibedakan menjadi empat macam, yakni *bandwidth, throughput, packet loss* dan *delay*.

6.1 Pengujian

Pada pengujian ini, desain arsitektur adalah seperti pada topologi yang direncanakan pada bab sebelumnya. Pengukuran dilakukan menggunakan aplikasi *Iperf* dan *bandwidth manager*.

6.1.1 Pengujian *Throughput*

Pengujian ini dilakukan untuk mengetahui berapa *throughput* dari *hardware* atau sistem yang telah dibangun. Pengujian ini menggunakan protokol TCP. Pada protokol TCP yang menjadi prioritas adalah reliable data yang dikirimkan, yakni dalam artian semua data harus terkirim 100%, bila ada paket data yang gagal terkirim, maka akan dilakukan pengulangan pengiriman dari sisi server. Pengujian *throughput* dilakukan dengan cara :

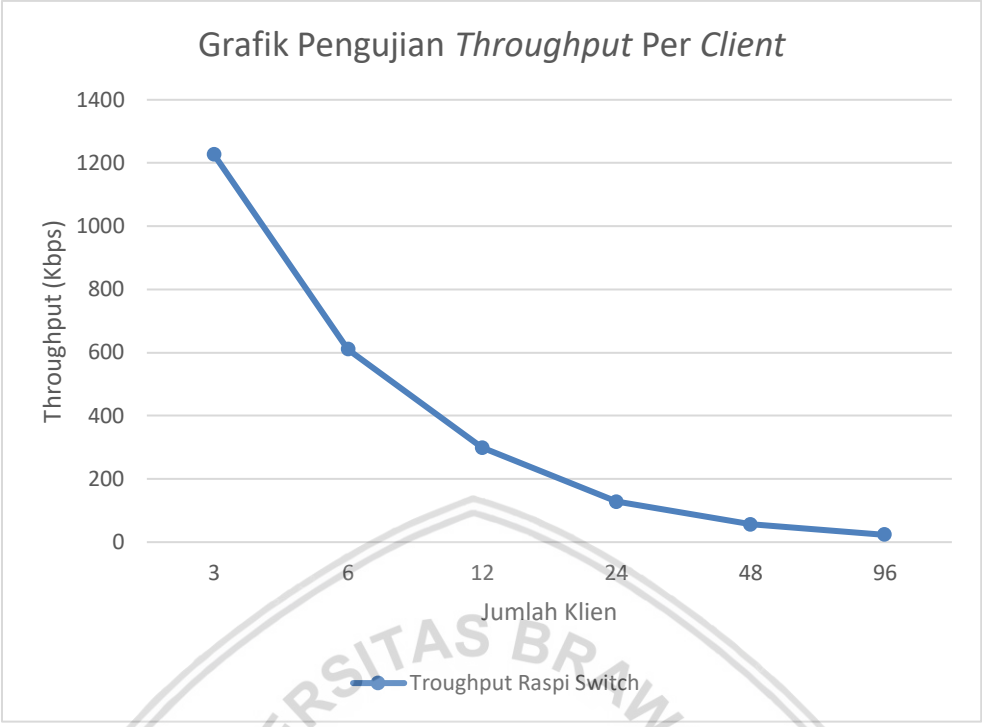
1. Merancang harware sesuai dengan topologi perencanaan
2. Aktifkan *controller*, *server* dan kedua *host*
3. Jalankan *Iperf –s* dan disisi server
4. Jalankan *Iperf –c 10.39.40.102 –P (jumlah host) –t 15* pada sisi *client.*
   (-P adalah membuat parallel *request*, sedangkan -t adalah waktu yang ditentukan)
5. Amati hasil keluaran yang ditunjukkan *Iperf*
6. Kesimpulan

Pengujian *throughput* dilakukan dengan skenario 4 klien, 8 klien, 12 klien dan 16 klien. Hasil yang didapat dari pengujian *throughput* tersaji pada tabel 6.1 di bawah ini.

Tabel 6.1 Hasil Pengujian *Throughput* Per Klient

Jumlah Klien	Rata-rata throughput (Kbps)
3	1226,67
6	610
12	299,17
24	127,92
48	55,84
96	23,02

Kemudian dari data-data tersebut dapat direpresentasikan menjadi grafik seperti pada gambar 6.1 berikut ini.



Gambar 6.1 Grafik *Throughput* Per Klien

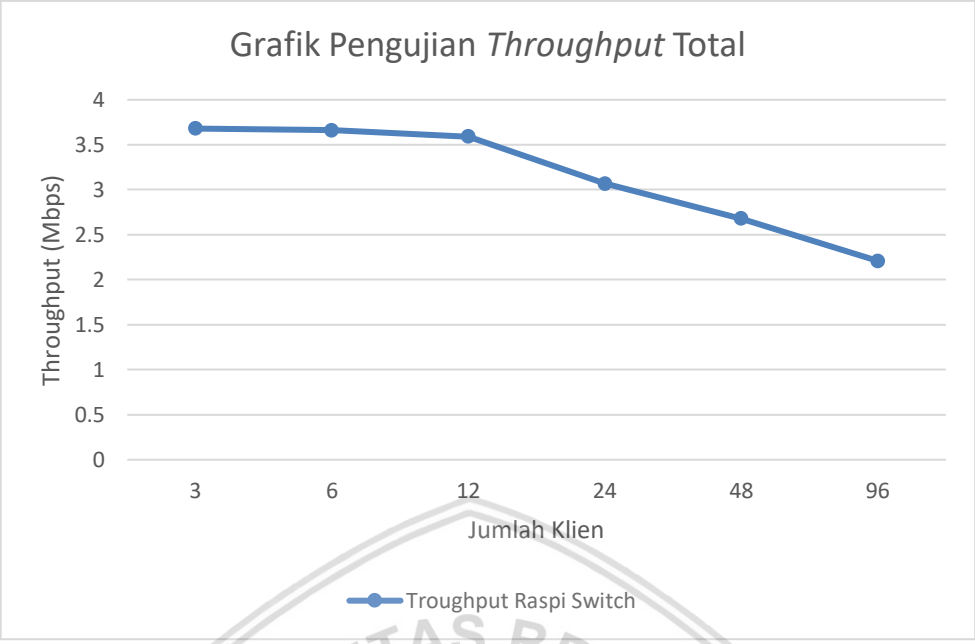
Dari tabel 6.1 dan grafik diatas, dapat dijelaskan bahwa untuk pengujian *throughput* pada 3 buah klien yang melakukan *request* bersamaan terhadap server, klien mendapatkan rata-rata *throughput* sebesar 1226,67 Kbps atau 1,2 Mbps. Selain itu dapat kita lihat juga, bahwa semakin sedikit klien yang meminta *request* terhadap server maka semakin besar ia mendapatkan *throughput,* dan sebaliknya semakin banyak user yang melakukan *request* terhadap server maka *throughput*nya pun yang akan didapatkan per klien juga semakin kecil dikarenakan harus dibagi ke seluruh klien yang melakukan *request*.

Dari tabel dan grafik diatas dapat disimpulkan bahwa fungsi dari switch openflow dapat berjalan dengan baik namun mendapatkan hasil yang relatif sangat kecil. Semakin banyak klien yang terhubung dan melakukan *request* terhada server maka *throughput* yang didapat juga semakin kecil dikarenakan harus dibagi ke seluruh klien yang melakukan *request*.

Tabel 6.2 Hasil Pengujian *Throughput* Total

Jumlah Klien	Total throughput (Mbps)
3	3,68
6	3,66
12	3,59
24	3,07
48	2,68
96	2,21

Kemudian dari data-data tersebut dapat direpresentasikan menjadi grafik seperti pada gambar 6.2 berikut ini.



Gambar 6.2 Grafik *Throughput* Total

Dari tabel 6.2 dan grafik diatas, dapat dijelaskan bahwa untuk pengujian *throughput* dari klien yang berjumlah 3 sampai dengan 96 klien, didapatkan hasil bahwa *throughput* total mengalami penurunan, dimulai dari 3,68 Mbps, 3,6 Mbps, 3,59 Mbps, 3,07 Mbps, 2,68 Mbps hingga 2,21 Mbps. Hal ini dapat diartikan bahwa semakin sedikit klien yang melakukan request data terhadap server, maka jumlah *throughput* semakin sempurna memenuhi *bandwidth* (lebar pita) maksimal yang tersedia oleh sistem, yakni sekitar 4 Mbps. Sedangkan bila semakin banyak klien yang melakukan request data terhadap server, seperti yang ditunjukkan dalam penelitian ini pengaksesan dari 96 klien menghasilkan grafik penurunan yang konstan. Penurunan *throughput* ini disebabkan oleh *bandwidth* atau lebar pita yang tersedia oleh sistem sebagian dipergunakan untuk proses komunikasi antara server dengan klien. Karena klien yang melakukan komunikasi begitu banyak, akhirnya lalu lintas jaringan pun menjadi semakin padat.

Dalam pengujian ini nilai 96 adalah jumlah klien maksimum yang dapat ditangani oleh sistem. Penguji telah melakukan skenario untuk 100 hingga 192 klien, namun hasil yang didapat sistem error dan tidak dapat mengirimkan data. Hal ini dapat disimpulkan bahwa 96 klien merupakan *saturation point* atau titik maksimal atau titik jenuh dari sistem.

6.1.2 Pengujian *Packet Loss*

Pengujian *packet loss* bertujuan untuk mengetahui berapa % *packet loss* dari *hardware* atau sistem yang telah dibangun ketika dilakukan *request* dari beberapa skenario yang akan dibangun.

Pengujian ini menggunakan protokol UDP. Didalam protokol UDP yang paling menjadi prioritaskan adalah kecepatan transfer, bukan pada reliable data yang dikirimkan. Pengujian ini dilakukan dengan skenario request dari 3 klien, 6

klien, 12 klien, 24 klien, 48 klien dan 96 klien. Pengujian *packet loss* dilakukan dengan cara :

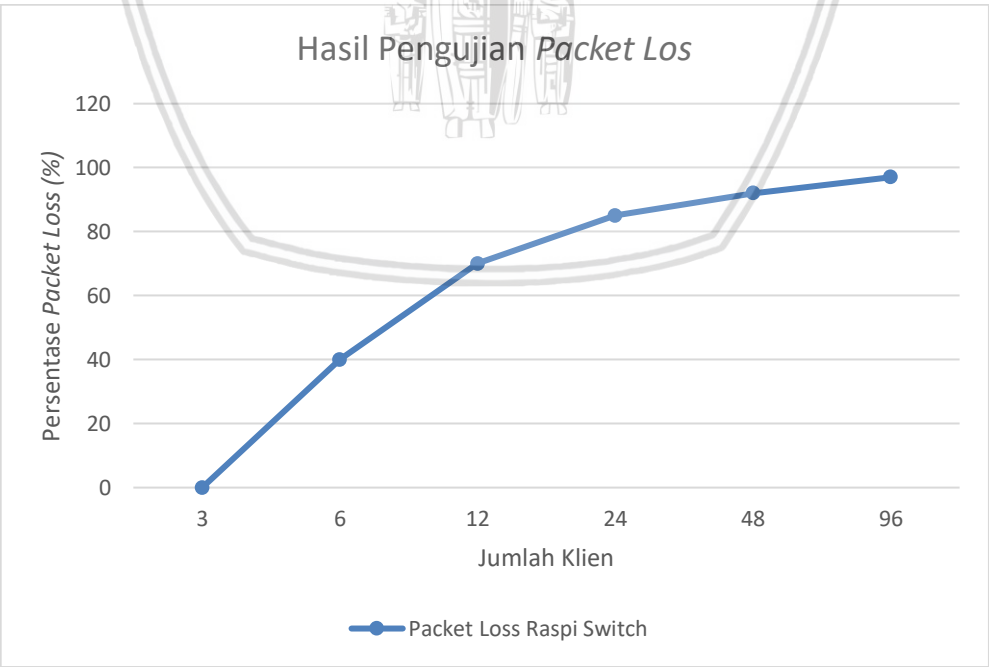
1. Merancang harware sesuai dengan topologi perencanaan
2. Aktifkan *controller*, *server* dan kedua *host*
3. Jalankan *Iperf –s* dan disisi server
4. Jalankan *Iperf –c 10.39.40.101 –P (jumlah klien) –u –t 15* pada sisi klient. (-P adalah membuat parallel *request*, -u yang berarti protokol UDP, sedangkan -t adalah waktu yang ditentukan)
5. Amati hasil keluaran yang ditunjukkan *Iperf*
6. Kesimpulan

Hasil yang didapat dari pengujian *packet loss* tersaji pada tabel 6.3 di bawah ini.

Tabel 6.3 Hasil Pengujian *Packet Loss*

Jumlah Klien	**Rata-rata *Packet Loss* (%)**
3	0
6	40
12	70
24	85
Jumlah Klien	**Rata-rata *Packet Loss* (%)**
48	92
96	97

Kemudian dari data-data tersebut dapat direpresentasikan menjadi grafik seperti pada gambar 6.3 berikut ini.



Gambar 6.3 Grafik *Packet Loss*

Dari tabel 6.3 dan grafik diatas, dapat dijelaskan bahwa untuk pengujian *packet loss* telah didapatkan hasil untuk pengaksesan jumlah klien muali dari 3, 6, 12, 24, 48 klien hingga 96 klien. Pada pengujian 3 buah klien yang melakukan *request* terhadap server, klien mendapatkan rata-rata *packet loss* 0%, yang berarti semua data dapat terkirimkan 100%. Seiring dengan bertambahnya klien yang melakukan *request* terhadap server, nilai *packet loss* yang terjadi semakin bertambah hingga mendekati 100%. Hal ini dikarenakan lalu lintas yang cukup padat antara server ke seluruh klien yang melakukan *request* dan juga karena paket yang dikirimkan oleh klien akan melebihi *bandwidth* dari sistem ini. Sehingga paket tersebut akan didrop dan menghasilkan *packet loss* yang signifikan.

6.1.3 Pengujian Delay

Pengujian delay bertujuan untuk mengetahui berapa delay dari sistem yang telah dibangun. Pengujian *delay* dilakukan dengan cara :

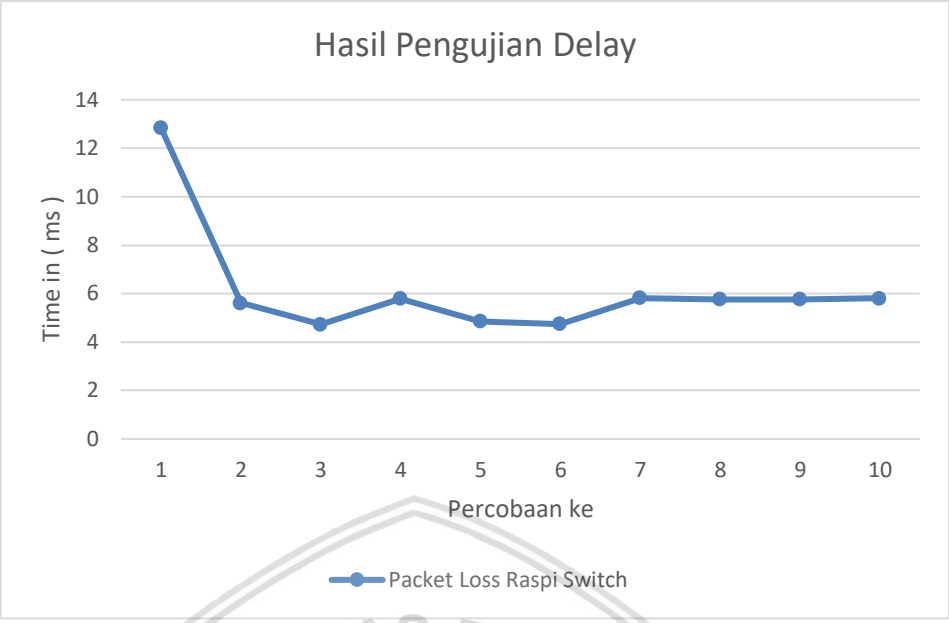
1. Merancang harware sesuai dengan topologi perencanaan
2. Aktifkan *controller*, *server* dan kedua *host*
3. Jalankan *Iperf –s* dan disisi server
4. Jalankan *Iperf –c 10.39.40.101 –c 10*
5. Amati hasil keluaran yang ditunjukkan *Iperf*
6. Kesimpulan

Pengujian ini dilakukan dengan skenario mengirimkan paket ICMP sebanyak 10 kali. Hasil yang didapat akan ditampilkan pada tabel 6.4 berikut ini.

Tabel 6.4 Hasil Pengujian Delay

Percobaan Ke	Rata-rata Delay (ms)
1	12,84
2	5,61
3	4,73
4	5,79
5	4,86
6	4,74
7	5,82
8	5,76
9	5,77
10	5,81

Dari tabel 6.4 diatas, dapat dijelaskan bahwa untuk pengujian *delay* hasil yang didapatkan dapat dibilang konstan, berkisar antara 4,73 ms – 12,84 ms yang berarti respon dari server dapat diterima oleh klien secara cepat, tidak ada packet icmp yang lost. Selanjutnya untuk melihat grafik *delay* dapat dilihat pada gambar 6.4 dibawah ini.



Gambar 6.4 Grafik *Delay* Raspi Switch OF

Dari tabel dan grafik diatas, percobaan yang pertama memiliki delay yang lebih lama dibandingkan dengan percobaan lainnya, hal ini dikarenakan pada paket ICMP (ping) yang pertama, terjadi proses ARP sehingga memerlukan waktu yang lebih lama. Pada paket yang kedua dan seterusnya tidak melakukan proses ARP dikarenakan informasi mengenai ARP sudah ada pada ARP cache.

Dari tabel dan grafik diatas, juga dapat disimpulkan bahwa fungsi dari switch openflow dapat berjalan dengan baik. Awal kali klient mengirimkan paket icmp ke server, server melakukan respon cukup tinggi yakni hingga 12 ms, sedangkan untuk percobaan ke-2 dan seterusnya, waktu yang diperlukan semakin tetap antara 4,73 ms – 5,82 ms.

6.2 Analisis

Analisis dilakukan setelah hasil dari percobaan didapatkan, analisis akan dijelaskan pada sub bab dibawah ini.

6.2.1 Analisis Throughput

Berdasarkan pengujian *throughput*, *throughput* per klien akan mengalami penurunan seiring dengan bertambahnya jumlah klien. Hal ini dikarenakan jika jumlah klien semakin banyak, maka bandwidth tersebut akan dibagi dengan jumlah klien yang sedang mengakses server tersebut sehingga *throughput* yang didapat oleh klien akan semakin kecil.

Sedangkan pada *throughput* total, didapatkan kesimpulan semakin sedikit klien yang melakukan request data terhadap server, maka jumlah *throughput* semakin sempurna memenuhi *bandwidth* (lebar pita) maksimal yang tersedia oleh sistem, yakni sekitar 4 Mbps. Sedangkan bila semakin banyak klien yang melakukan request data terhadap server, seperti yang ditunjukkan dalam penelitian ini pengaksesan dari 96 klien menghasilkan grafik penurunan yang konstan. Penurunan *throughput* ini disebabkan oleh *bandwidth* atau lebar pita

yang tersedia oleh sistem sebagian dipergunakan untuk proses komunikasi antara server dengan klien. Karena klien yang melakukan komunikasi begitu banyak, akhirnya lalu lintas jaringan pun menjadi semakin padat.

Dalam pengujian ini nilai 96 adalah jumlah klien maksimum yang dapat ditangani oleh sistem. Penguji telah melakukan skenario untuk 100 hingga 192 klien, namun hasil yang didapat sistem error dan tidak dapat mengirimkan data. Hal ini dapat disimpulkan bahwa 96 klien merupakan *saturation point* atau titik maksimal atau titik jenuh dari sistem.

6.2.2 Analisis Packet Loss

Berdasarkan pengujian *packet loss*, *packet loss* akan naik secara signifikan ketika jumlah klien lebih dari 3. Hal ini dikarenakan ketika jumlah klien lebih dari 3, maka paket yang dikirimkan oleh klien akan melebihi *bandwidth* dari sistem ini. Sehingga paket tersebut akan didrop dan menghasilkan *packet loss* yang signifikan. Selain itu hal ini juga dipengaruhi oleh lalu lintas yang cukup padat antara server ke seluruh klien yang melakukan *request*.

6.2.3 Analisis Delay

Pada pengujian delay, percobaan yang pertama memiliki delay yang lebih lama dibandingkan dengan percobaan lainnya, hal ini dikarenakan pada paket ICMP (ping) yang pertama, terjadi proses ARP sehingga memerlukan waktu yang lebih lama. Pada paket yang kedua dan seterusnya tidak melakukan proses ARP dikarenakan informasi mengenai ARP sudah ada pada ARP cache.

BAB VII KESIMPULAN

Pada bab ini membahas tentang kesimpulan dari penelitian yang telah dilakukan yaitu "Implementasi *Switch Openflow Software-based* Dengan Memanfaatkan Raspberry Pi Untuk Infrastruktur SDN" dan saran-saran untuk pengembangan skripsi atau penelitian lebih lanjut.

7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis dari implementasi sistem ini, dapat ditarik kesimpulan dan saran dari penelitian ini, yaitu sebagai berikut.

1. Sistem dirancang dengan 2 buah raspi yang telah ditambahkan modul USB to Ethernet sebanyak 10 unit yang dipasangkan pada raspi masing masing 3 unit, sedangkan sisanya untuk server, *controller* dan 2 buah *host*. Perancangan ini dimulai dari membuat skenario topologi, dimana pada penelitian ini penulis menggunakan topologi linear. Setelah itu perancangan sistem openflow pada raspi, yakni dengan memilih OS Linux rasbian dan *software* OpenvSwitch 2.30 pada raspi agar dapat berfungsi sebagai switch openflow. Sedangkan langkah terahir adalah perancangan *controller,* server dan *host* yang penulis tempatkan pada virtual machine di macbook. *Controller* dalam penelitian kali ini dirancang menggunakan floodlight ber OS-kan ubuntu 14.04 LTS. Sedangkan pada sisi server dan *host* penulis menggunakan OS Linux bunsenlabs karena sistemnya yang sangat ringan.
2. Implementasi dilakukan dengan cara merakit semua komponen sesuai dengan topologi dan skema diagram sistem yang telah dirancang sebelumnya. Selanjutnya melakukan installasi *operating-system* pada Raspberry dan juga *software* OpenvSwitch beserta dengan konfigurasi IP, *bridge, port* dan pengarahan *tcp controller port*. Langkah selanjutnya adalah melakukan implementasi *virtual machine* untuk *controller, host* dan server dimulai dari penginstallan *operating-sistem* sesuai dengan perancangan sebelumnya, kemudian penambahan floodlight *controller, Iperf* dan *Bandwidth monitoring.*
3. Hasil yang didapatkan dari pengujian switch openflow *software-based* pada raspberry ini berdasarkan pengujian *throughput, throughput* akan mengalami penurunan seiring dengan bertambahnya jumlah klien. Hal ini dikarenakan jika jumlah klien semakin banyak, maka bandwidth tersebut akan dibagi dengan jumlah klien yang sedang mengakses server tersebut sehingga throughput yang didapat oleh klien akan semakin kecil. Berdasarkan pengujian *packet loss*, *packet loss* akan naik secara signifikan ketika jumlah klien lebih dari 4. Hal ini dikarenakan ketika jumlah klien lebih dari 4, maka paket yang dikirimkan oleh klien akan melebihi *bandwidth* dari sistem ini. Sehingga paket tersebut akan didrop dan menghasilkan packet loss yang signifikan. Terahir berdasarkan pada pengujian *delay*, percobaan yang pertama memiliki *delay* yang lebih lama dibandingkan dengan percobaan lainnya, hal ini dikarenakan pada paket ICMP (ping) yang pertama, terjadi

proses ARP sehingga memerlukan waktu yang lebih lama. Pada paket yang kedua dan seterusnya tidak melakukan proses ARP dikarenakan informasi mengenai ARP sudah ada pada ARP *cache.*

7.2 Saran

Berdasarkan hasil penelitian perancangan dan pengujian switch openflow pada raspberry pi, saran yang dapat diberikan penulis untuk penelitian selanjutnya adalah sebagai berikut:

1. Agar memilih adapter USB 3.0 Gigabit Ethernet yang asli, sehingga diharapkan akan mendapatkan hasil yang lebih baik.
2. Pengujian dilakukan dengan lebih luas, *big switch* yaitu dengan cara didalam sebuah Raspi switch OF akan mencabang lagi ke beberapa switch OF dibawahnya, baru kemudian terhubung dengan *host-host* dan server, sehingga didapatkan hasil untuk dilakukan analisis lebih mendalam.
3. Dikembangkan lagi dengan skala lebih besar dengan Raspi switch 10 buah, dan dengan skenario beberapa topologi sistem, sehingga bisa didapatkan hasil untuk dilakukan analisis lebih mendalam berupa metode routing yang akan digunakan ataupun yang lain sebagainya.

DAFTAR PUSTAKA

Abdillah, N., 2016. *Analisis Performa Arsitektur Software Defined Network dengan Openflow Pada Mikrotik RB 750*. Yogyakarta: STMIK Amikom.

Cahyadi, A., 2017. *Pengantar SDN.* [e-book] GitBook. [online] Tersedia di: <https://eueung.gitbooks.io/buku-komunitas-sdn-rg/content/pengantar_sdn/README.html> [Diakses 24 September 2017].

Floodlight Controller, 2017. *Installation Guide – Project Controller – Project Floodlight.* [online] Tersedia di: <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Installation+Guide> [Diakses 12 November 2017].

Iperf, 2018. *Bandwidth measurement tools.* [online] Tersedia di: <https://iperf.fr/> [Diakses 12 Mei 2018].

Kartadie, R., 2016. *Mikrotik RB750 Router Board Sebagai Alternatif Switch OpenFlow Software Based.* Tulungagung: Jurnal SIMETRIS, Vol. 7 No. 2.

Kartadie, R., 2014. *Prototipe Infrastruktur Software-Defined Network Dengan Protokol OpenFlow Menggunakan UBUNTU Sebagai Kontroler*. Yogyakarta: Jurnal DASI, Vol. 15 No. 1.

Khoerul Anam, dkk. 2017. *Arsitektur SDN vs Arsitektur Tradisional*. Yogyakarta: STMIK Amikom.

MikrotikID, 2016. *Arsitektur SDN beserta Komponen dan Interaksinya.* [pdf] Seminar Mikrotik Indonesia. Tersedia di: <https://mum.mikrotik.com/presentations/ID15/presentation_2611_1444662871.pdf> [Diakses 25 September 2017].

Naous, J., 2008. *Implementing an OpenFlow Switch on the NetFPGA Platform.* California: Stanford University.

Ngonoo Situs Antu Kudet, 2017. *5 Distro Linux Paling Ringan, Cocok Untuk Laptop Dengan Prosesor Intel Atom*. [online] Tersedia di: <https://ngonoo.com/2017/05/200330/5-distro-linux-paling-ringan-cocok-untuk-laptop-dengan-prosesor-intel-atom/> [Diakses 19 November 2017].

Opennetworking, 2017. *Software Defined Networking (SDN) Definition*. [online] Tersedia di: <https://www.opennetworking.org/sdn-resources/sdn-definition> [Diakses 24 September 2017].

Open Networking Foundation ONF, 2013. *SDN Architecture Overview*. [online] Tersedia di: <http://www.opennetworking.org/wp-content/uploads/2013/02/SDN-architecture-overview-1.0.pdf> [Diakses 23 September 2017].

OpenvSwitch, 2017. *Openvswitch ovs.* [online] Tersedia di: <https://github.com/openvswitch/ovs> [Diakses 12 November 2017].

Fadhil, M., 2017. *Software Defined Network, karena layer dan topologi tidak penting lagi.* [online] Tersedia di: <http://blog.randisunarsa.web.id/?p=610> [Diakses 28 September 2017].

Raspberry Pi Organization, 2017. *Raspberry Pi 3 Model B*. [online] Tersedia di: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b> [Diakses 27 September 2017].

Rikie, K. dan Barka, S., 2015. *Uji Performa Implementasi Software Based OpenFlow Switch Berbasis OpenWrt Pada Infrastruktur Software Defined Network.* Yogyakarta: STMIK Amikom.

Scott Shenker, 2011. *The Future of Networking, and the Past of Protocols.* [online] Tersedia di: <http://opennetsummit.org/archives/ oct11/shenker-tue.pdf> [Diakses 23 September 2017].

SDx Central | Trusted News and Resource for SDx, SDN & NFV, 2017. *What is floodlight controller.* [online] Tersedia di: <https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-floodlight-controller/> [Diakses 06 November 2017].

Superuser the OpenStack User Publication, 2017. *Understanding Open vSwitch, an OpenStack SDN component.* [online] Tersedia di: <http://superuser.openstack.org/articles/openvswitch-openstack-sdn/> [Diakses 16 Oktober 2017].